

The image is a large, symmetrical, abstract graphic composed of the letters 'S' and 'Y' arranged in a pattern that resembles a stylized 'Y' or a complex geometric shape. The letters are black on a white background. The central vertical column is composed of 'Y's, while the horizontal and diagonal branches are composed of 'S's. The overall shape is roughly triangular, with the 'Y' characters forming the central spine and the 'S' characters forming the outer edges and internal structure. The pattern is highly regular and repetitive, creating a sense of depth and complexity.

SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	QQQQQQ	ZZZZZZZZZZ
SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FFFFFFFFFF	QQQQQQ	ZZZZZZZZZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SSSSSSS	YY	YY	SSSSSSS	DD	EE	FFFFFFFFF	QQ	ZZ
SSSSSSS	YY	YY	SSSSSSS	DD	EE	FFFFFFFFF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SS	YY	YY	SS	DD	EE	FF	QQ	ZZ
SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FF	QQQQ	ZZZZZZZZZZ
SSSSSSSS	YY	YY	SSSSSSSS	DDDDDDDD	EEEEEEEEEE	FF	QQQQ	ZZZZZZZZZZ

```

SSSSSSSS DDDDDDDD LL
SSSSSSSS DDDDDDDD LL
SS        DD        LL
SS        DD        LL
SS        DD        LL
SS        DD        LL
      SSSSSS DD        LL
      SSSSSS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
                SS DD        LL
SSSSSSSS DDDDDDDD LLLLLLLLLL
SSSSSSSS DDDDDDDD LLLLLLLLLL

```

[illegible]

Version: 'V04-000'

COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
ALL RIGHTS RESERVED.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
TRANSFERRED.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
CORPORATION.

DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.

FACILITY: VAX/VMS System Macro Libraries

ABSTRACT:

This file contains the SDL source for all operating system control
blocks, from Q to Z. That is, all control blocks from QAA to ZZZ.

ENVIRONMENT:

n/a

AUTHOR: The VMS Group

CREATION DATE: 1-Aug-1976

MODIFIED BY:

V03-125 ROW0410 Ralph O. Weber 6-AUG-1984
Define a UCBSL_STS bit indicating VAXcluster state transition
processing in progress, UCBSV_CLOTRAN. This bit is used to
force mount verification to execute after (or as a part of) a
VAXcluster state transition.

V03-124 ACG0441 Andrew C. Goldstein, 2-Aug-1984 15:16
Add CLUSLOCK flag to VCB, DISMOUNT flag to UCB

V03-123 TCM0005 Trudy C. Matthews 23-Jul-1984 16:50

Add RPBSB_CTRLTR to SRPBDEF.

- V03-122 ACG0438 Andrew C. Goldstein, 23-Jul-1984 15:16
Add VCA structures for cache interlocks
- V03-121 MMD0314 Meg Dumont, 19-Jul-1984 12:23
Add \$XGDEF to LIB.ML
- V03-120 ROW392 Ralph O. Weber 19-JUL-1984
Add UCBSW_DEVSTS bit UCBSV_MSCP_W RTP which is the inclusive or
of all the various MSCP write protection bits.
- V03-119 LY0501 Larry Yetto 6-JUL-1984 13:11
Add subfields to UCBSL_MEDIA_ID
- V03-118 ROW0379 Ralph O. Weber 21-JUN-1984
Add UCBSW_2P_MSCPUNIT to provide for future implementation of
dual-path UDA support via MSCP unit number switching in the
MSCP class driver.
- V03-117 ROW0357 Ralph O. Weber 1-MAY-1984
Add UCBSV_MNTVERPND in UCBSL_STS. This bit will be set along
with UCBSV_MNTVERIP when it is necessary to stall a busy
device due to loss of quorum. Also add UCBSV_MSCP_PKACK in
UCBSW_DEVSTS for MSCP devices. This will be used by the disk
class driver to indicate that a IOS_PACKACK operation is in
progress.
- V03-116 CDS0012 Christian D. Saether 18-Apr-1984
Add wcb\$w_noacclck, remove wcb\$w_acclkid.
- V03-115 GRR4115 Gregory R. Robert 12-Apr-1984
Split \$SMBDEF into symbiont services and print symbiont
sections. Move latter into \$PSMDEF.
- V03-114 LJK0265 Lawrence J. Kenah 9-Apr-1984
Add SHL_SIZE cell and OLD_SHL_SIZE constant to \$SHLDEF
to allow image activator and ANALYZE /IMAGE to handle
shareable image list elements of different sizes.
- V03-113 LMP0221 L. Mark Pilant, 7-Apr-1984 13:13
Add a pointer to the ORB in the UCB. Also remove the
definitions, in the UCB, of the owner UIC, volume
protection, and the ACL queue segment list head.
- V03-112 ROW0337 Ralph O. Weber 7-APR-1984
Remove UCBSL_CANLINK. This field is no longer used.
- V03-111 MHB0118 Mark Bramhall 26-Mar-1984
Add UAF\$V_DISRECONNECT.
- V03-110 MIRO370 Michael I. Rosenblum 20-Mar-1984
Add definitions for ITY\$V_ST_TABRIGHT and CTSLOW
Add UCB cell to store the cursor positioning from the
last read.

V03-109 SSA0020 Stan Amway 20-Mar-1984
Moved UCB\$W_QLEN from disk/tape extension to main UCB.
Changed UCB\$W_DCCB in disk/tape extension to UCB\$L_DCCB.

V03-108 ROW0325 Ralph O. Weber 19-MAR-1984
Remove UCB\$L_MVIOQFL and UCB\$L_MVIOQBL; these fields are no longer used. Add UCB\$Q_MSCP_RESV, eight bytes of space reserved for use by new MSCP features implemented during the Version 4 life-time. Add VCB\$L_SHAD_RESV to the standard disk VCB and VCB\$Q_SHDM_RESV to the shadow set member VCB, both fields provide reserved space for implementation of volume shadowing during the Version 4 life-time.

V03-107 JWT0163 Jim Teague 10-Mar-1984
Enlarge name fields in shareable image list (\$SHLDEF)

V03-106 ACG0400 Andrew C. Goldstein, 10-Mar-1984 2:01
Add quota cache structures for cluster-wide quota cacheing

V03-105 WHM0001 Bill Matthews 02-Mar-1984
Added field SLV\$A_SYSVECS(address of vectors in SYS.EXE) to \$SLVDEF.

V03-104 CDS0011 Christian D. Saether 8-Mar-1984
Add WRITE_TURN flag to WCB.

V03-103 PRD0072 Paul R. DeStefano 27-Feb-1984
Added SB\$L_CSB (link to newest Cluster System Block) to \$SBDEF.

V03-102 ROW0316 Ralph O. Weber 27-FEB-1984
Change UCB\$V_TU_OVRSEQCHK to UCB\$V_TU_OVRSQCHK in UCB\$W_DEVSTS. Add a fourth type of VCB for shadow set members. Add fields to the disk VCB for shadow set and enhanced mount verification handling.

V03-101 SSA0011 Stan Amway 27-Feb-1984
Changed UCB\$L_DCCB in disk/tape UCB extension to UCB\$W_DCCB.
Moved UCB\$W_QLEN to disk/tape UCB extension.
Deleted UCB\$L_WRTCNT.

V03-100 MMD0244 Meg Dumont, 24-Feb-1984 14:59
Add support for VCB\$V_FIL_ACCESS

V03-099 SSA0010 Stan Amway 14-Feb-1984
Added UCB\$V_DATACACHE as a disk-specific bit in UCB\$W_DEVSTS.
Added UCB\$L_DCCB in disk/tape UCB extension.

Added UCB\$L_WRTCNT & UCB\$W_QLEN to support MONITOR disk class.
(Done on behalf of Tom Cafarella.)

V03-098 ROW0303 Ralph O. Weber 10-FEB-1984
Add definition of UCB\$M_AST_ARMED defined as the sign bit of UCB\$W_DIRSEQ.

V03-097 ROW0300 Ralph O. Weber 9-FEB-1984

Add UCBSV_MSCP_FLOVR, a bit which a MSCP driver toggles (changes the state of) whenever a device successfully moves from one controller to another.

- V03-096 ROW0297 Ralph O. Weber 7-FEB-1984
Add UCBSV_TU_SEQNOP a device dependent status bit which the tape class driver sets when the wait count is bumped due to a sequential nop operation being in progress.
- V03-095 PCG0001 Peter George 06-feb-1984
Add UAFSV_DISREPORT.
- V03-094 LMP0188 L. Mark Pilant, 4-Feb-1984 11:19
Add a classification block to the VCB.
- V03-093 TMK0004 Todd M. Katz 04-Feb-1984
Add a NI device extension to \$UCBDEF.
- V03-092 KPL0100 Peter Lieberwirth 2-Feb-1984
Lengthen RPB_BOOTNDT field to a word. Leave byte definition as well. Only newly-defined FLAGS byte need move as a result, it shifts left one byte.
- V03-091 ROW0282 Ralph O. Weber 14-JAN-1984
Add UCBSV_SUPMMSG bit to UCBSL_STS. When set, this bit suppresses success status messages from mount verification. Add UCBSL_WAIT_CDDb to the MSCP UCB extension. This field points to a CDDb which is waiting for mount verification to complete before beginning single CDRP processing.
- V03-090 ACG0385 Andrew C. Goldstein, 9-Jan-1984 17:07
Replace \$UAFDEF (authorization file) with new V4 format
- V03-089 LMP0177 L. Mark Pilant, 7-Dec-1983 10:22
Add an ACL queue listhead to the UCB.
- V03-088 CDS0010 Christian D. Saether 6-Dec-1983
Add VCB\$V_NOSHARE flag.
- V03-087 SRB0106 Steve Beckhardt 6-Dec-1983
Made several changes to \$RSBDEF.
- V03-086 ROW0256 Ralph O. Weber 16-NOV-1983
Move the general disk UCBSW_DEVSTS bit definitions from being a UCBSL_DPC subfield to being a UCBSW_DEVSTS subfield. Also redo the MSCP and TU device status bit definitions to accomodate these recently discovered disk device status bits.
- V03-085 ROW0255 Ralph O. Weber 14-NOV-1983
Restore UCBSW_EC1 and UCBSW_EC2 symbols which were lost during the preparation of ROW0253.
- V03-084 TMK0003 Todd M. Katz 12-Nov-1983
Lets try again and this time fix both TMK0002 and ROW0253. My previous attempt doubly defined the symbol UCBSK_2P_LENGTH. Replace one of the definitions with local symbol #2P_LENGTH.

- V03-083 TMK0002 Todd M. Katz 12-Nov-1983
Fix ROW0253 and system build by defining UCB\$K_2P_LENGTH.
- V03-082 ROW0253 Ralph O. Weber 11-NOV-1983
Make changes to the UCB to accomidate more class driver fields becoming publicly available. Make DEVSTS definitions for class driver bits. Add 2P synonyms for all dual-path fields. Restructure disk and tape specific sections of the device dependent UCB to add class driver fields and to overlay some unused local-disk fields with these class driver fields.
- V03-081 TMK0001 Todd M. Katz 26-Oct-1983
Add UAF\$J_TQUOTA within SUAFDEF.
- V03-080 CDS0009 Christian D. Saether 14-Oct-1983
Remove VCB\$ALLOCLKID. Add various VCB and RVT fields to support volume activity blocking for rebuild.
- V03-079 CDS0008 Christian D. Saether 10-Oct-1983
Add VCB\$ALLOCLKID.
- V03-078 JSV0411 Joost Verhofstad 27-SEP-1983
Add UCB\$W_JNL_PROT
- V03-077 GRR0005 Gregory R. Robert 26-Sep-1983
Added \$SMBDEF, public symbiont definitions.
- V03-076 CWH3076 CW Hobbs 10-Sep-1983
Add a RPBSB_FLAGS field to \$RPBDEF, and define a bit RPBSV_NOSYS\$DISK, which means that the boot volume is no longer present (e.g. for S/A BACKUP from console).
- V03-075 ROW0220 Ralph O. Weber 7-SEP-1983
Switch bit positions of the Phase 1 in progress and Phase 2 in progress flags in the Replacement and Caching Table structure definition, \$RCTDEF. This causes the definition to conform with the DSDF specification, from which it is derived.
- Also define UCB\$V_LCL_VALID in UCB\$S_STS. This bit being set indicates that one of the reasons a disk is volume valid is a PACKACK operation performed on the local node.
- V03-074 CDS0007 Christian D. Saether 24-Aug-1983
Add RVT\$T_VLSLCKNAM to the RVT.
- V03-073 ROW0211 Ralph O. Weber 16-AUG-1983
Add two LENGTH symbols to \$UCBDEF. UCB\$K_LCL_DISK_LENGTH is the length of a device-independent UCB for a local disk. UCB\$K_LCL_TAPE_LENGTH is the length of a device-independent UCB for a local magnetic tape. (Someday there will be different LENGTH symbols for remote disks and tapes.)
- Also make UCB\$W_BCR have a longword form for the convenience of DBDRIVER.

V03-072 ROW0204 Ralph O. Weber 5-AUG-1983
Move UCB\$\$_CPID from its current overlay with UCB\$\$_DUETIM to
a new overlay with UCB\$\$_LOCKID.

V03-071 KTA3072 Kerbey T. Altmann 02-Aug-1983
Add UCB\$\$_ONLCNT - count of number of ONLINES to
a disk or tape.
Change location of RVT\$\$_STRUCLKID added by CDS0006.

V03-070 CDS0006 Christian D. Saether 2-Aug-1983
Remove RVX structure.
Add RVT\$\$_STRUCLKID.

V03-069 NPK3029 N. kronenberg 29-JUL-1983
Redefine the SCSSC_ST status codes to conform to
the latest SCA format (error type*8+severity.)

V03-068 LY0401 Larry Yetto 29-JUL-1983 12:15:59
Add UCB\$\$_JNL_BTSEQNO

V03-067 JLV0283 Jake VanNoy 28-JUL-1983
Correct fill problem in UCBDEF.

V03-066 JSV0367 Joost Verhofstad 28-JUL-1983
Change journal name length to 18

V03-065 MMD0190 Meg Dumont, 28-Jul-1983 9:45
Changed bit in VCBDEF from AUTO to NOAUTO to make mag tape
AVL/AVR consistent between DCL and MOUNT system service

V03-064 LY0398 Larry Yetto 28-JUL-1983
Add UCB\$\$_JNL_WCBFL and UCB\$\$_JNL_WCBBL. Remove
VCB\$\$_JNL_WCBFL and VCB\$\$_JNL_WCBBL.

V03-063 CDS0005 Christian D. Saether 28-Jun-1983
Add VCB\$\$_VOLCKID, VCAS\$_EXTCLKID, and VCAS\$_FIDCLKID.

V03-062 MIR0052 Michael I. Rosenblum 27-Jun-1983
Add missing definitions for UCB\$\$_RTT_BANDXCL and BANDXMSK

V03-061 MIR0051 Michael I. Rosenblum 24-Jun-1983
Add \$TTYUCBDEF to this module. Also note this Definition
Must follow \$UCBDEF as it gets a local symbol defined
in \$UCBDEF.

V03-060 CDS0004 Christian D. Saether 23-Jun-1983
Add VCB\$\$_VOLCKNAM to end of vcb instead of in middle.

V03-059 RLRDPATH1 Robert L. Rappaport 23-Jun-1983
Add UCB\$\$_DP_ALTUCB to Dual Path section of UCB.

V03-058 CDS0003 Christian D. Saether 23-Jun-1983
Add VCB\$\$_VOLCKNAM field.

V03-057 WMC0055 Wayne Cardoza 21-Jun-1983
Increase number of vector pages in SGNDEF

V03-056 PRB0198 Paul R. Beck 13-JUN-1983 13:28
Add RUCBDEF and RUHDEF (formerly defined in [RUF.SRC]RUF.SDL)

V03-055 LY0381 Larry Yetto 13-JUN-1983 07:57:45
Add UCB\$JL_JNL_RC

V03-054 WMC0054 Wayne Cardoza 29-May-1983
New protection and spare fields in SLVDEF.

V03-053 MLJ0113 Martin L. Jack 27-May-1983
Add UAF\$B_QUEPRI.

V03-052 RLRDPATH Robert L. Rappaport 25-May-1983
Create a new UCB extension for Dual Ported Devices that
resides after the errorlogging extension and before
the DISK_UCB_EXTENSION. Place in it UCB\$JL_DP_DDB and
UCB\$JL_DP_LINK.

V03-051 LY0375 Larry Yetto 24-MAY-1983 15:44:29
Add cluster write Q list head to UCB journal extension.
Also added status bits DIRENTER and CHKVAL in \$RSBDEF.

V03-050 KTA3052 Kerbey T. Altmann 24-May-1983
Updated SCSDEF for split up of layers.

V03-049 LY0366 Larry Yetto 18-MAY-1983 17:08:36
Add UCB\$V_JNL_UNMAST, UCB\$JL_JNL_RMBLK, UCB\$JL_JNL_ACBM,
and UCB\$JL_JNL_LSEQNO

V03-048 JLV0253 Jake VanNoy 18-MAY-1983
Add symbols to \$STASTDEF.

V03-047 MMD0151 Meg Dumont, 26-Apr-1983 9:06
Add VCB\$B_LBLCNT to the MTAACP portion of VCBDEF

V03-046 LY0356 Larry Yetto 21-APR-1983 08:16:48
Add UCB\$JL_JNL_FAILQFL and UCB\$JL_JNL_FAILQBL

V03-045 MSH0004 Maryann Hinden 14-Apr-1983
Add SPNBSW_REFC.

V03-044 JWH0208 Jeffrey W. Horn 12-Apr-1983
Add SLV\$B_PROT and SLV\$T_FACILTY to \$SLVDEF.

V03-043 TCM0004 Trudy C. Matthews 12-Apr-1983
Add UCB\$JL_LOCKID to \$UCBDEF.

V03-042 JWT0104 Jim Teague 29-Mar-1983
Add UAF\$T_CLITABLES field to \$UAFDEF.

V03-041 JSV0201 Joost Verhofstad 28-MAR-1983
Add monitor counts in UCB for journals

V03-040 ROW0171 Ralph O. Weber 25-MAR-1983
Extend UCB\$JL_DEVCHAR to a quadword, UCB\$Q_DEVCHAR.

Also create symbol for second portion of the quadword,
UCB\$\$_DEVCHAR2.

V03-039 SRB0072 Steve Beckhardt 25-Mar-1983
Added some fields to \$RSBDEF.

V03-038 MSH0003 Maryann Hinden 25-Mar-1983
Delete SPNB\$_HEADER definition, add SPNB\$_HDRS12.

V03-037 STJ3075 Steven Jeffreys 25-Mar-1983
- Add VCB\$_ERASE and VCB\$_NOHIGHWATER.

V03-036 MMD0108 Meg Dumont, 11-Mar-1983 12:32
Add defs to VL1DEF for handling of VOL1 owner id field
and added VL2DEF to handle the VOL2 label.

V03-035 MMD0106 Meg Dumont, 10-Mar-1983 9:55
Add fields in VCB for AVR, AVL and the VOL2, HDR4
additions to MTAACP

V03-034 SRB0069 Steve Beckhardt 9-Mar-1983
Removed SYSNAM status bit from \$RSBDEF.

V03-033 LY0316 Larry Yetto 07-mar-1983
Add WCB\$_JDB field to WCB\$_JNL_STAT

V03-032 WMC0001 Wayne Cardoza 06-Mar-1983
Add UAF\$_MAXDETACH

V03-031 JSV0161 Joost Verhofstad 28-FEB-1983
Add VCB\$_JNLIOCNT

V03-030 RLRMXBCNT Robert L. Rappaport 24-Feb-1982
Add UCB\$_MAXBCNT.

V03-029 MSH0002 Maryann Hinden 24-Feb-1983
Add \$SPNBDEF.

V03-028 ROW0139 Ralph O. Weber 18-FEB-1983
Move JNL_CLS and JNL_SLV to UCB\$_DEVSTS. Overlay UCB\$_PDT
with UCB\$_JNL_MCSID. Remove UCB\$_JNL_CHAR, UCB\$_JNL_CDT,
UCB\$_SCSFC, and UCB\$_SCSBL journal UCB extension.

V03-027 RSH0005 R. Scott Hanna 10-Feb-1983
Added \$RDIDEF which defines the rights database
identifier block offsets.

V03-026 TCM0003 Trudy C. Matthews 9-Feb-1983
Added new VMB input flags to \$RPBDEF: AUTOTEST and CRDTEST.

V03-025 SRB0065 Steve Beckhardt 21-Jan-1983
Added new resource CLUSTAN to \$RSNDEF.

V03-024 DWT0067 David W. Thiel 20-Jan-1983
Add \$SPPBDEF. Add fields to \$SBDEF.

V03-023 CDS0002 Christian D. Saether 27-Dec-1982
Move WCB\$\$_ACCLKID to avoid ambush by inexplicit assumptions.

V03-022 STJ3045 Steven T. Jeffreys 16-Dec-1982
Add \$SLVDEF macro definition.

V03-021 SRB0057 Steve Beckhardt 16-Dec-1981
Added some new fields and reordered others in \$RSBDEF
for distributed lock manager support.

V03-020 CDS0001 Christian D. Saether 9-Dec-1982
Add WCB\$\$_ACCLKID.

V03-019 ACG0303 Andrew C. Goldstein, 9-Dec-1982 15:13
Add FILL attribute to extraneous names

V03-018 NPK3010 N. Kronenberg 15-Nov-1982
Modify \$SBDEF to add node name, hardware type and version,
and DDB pointer. Add \$SBDEF and \$SYSAPDEF.

V03-017 MMD0001 Meg Dumont, 11-Nov-1982 14:42
Add bit inMODE field of VCBDEF to allow users to
enable EOT handling in the MTAACP.

V03-016 TCM0002 Trudy C. Matthews 31-Oct-1982
Add new field to Restart Parameter Block: RPB\$\$_BADPGS.

V03-015 KTA0017 Kerbey T. Altmann 21-Oct-1982
Add new fields to SCS system block.

V03-014 CWH0014 CW Hobbs 20-Oct-1982
Add some warnings to \$WSLDEF about strange constants

V03-013 JSV0081 Joost Verhofstad 8-Oct-1982
Add WCB\$\$_JNL_PUIC

V03-012 SRB0054 Steve Beckhardt 6-Oct-1982
Added new resource for SCS waits in RSNDEF.

V03-011 JSV0068 Joost Verhofstad 22-Sep-1982
Add some journaling specific WCB fields and change the fixed
constants produced by conversion routines from MDL, into
computed constants for the WCB and VCB.

V03-010 ROW0122 Ralph O. Weber 12-SEP-1982
Work over the UCB definition. Change machined SDL into
human senseable SDL. Extend UCBSW_ST\$ to a longword,
defining UCBSL_ST\$. Add a spare word for alignment after
UCBSW_DEVST\$. Move UCBSL_DEVDEPN2 next to UCBSL_DEVDEPEND
and create UCBSQ_DEVDEPEND.

V03-009 JSV0063 Joost Verhofstad 09-Sep-1982
Change names of symbols that existed in V3.0 and
are used for journaling and have changed. This is
to allow builds of journaling facilities on both

the latest system and V3.0 (for early field test)

V03-008	MSH0001	Maryann Hinden	09-Sep-1982
	Add UAS definitions.		
V03-006	JSV0031	Joost Verhofstad	27-Jul-1982
	Add some UCB and WCB fields for journaling		
V03-005	JSV0022	Joost Verhofstad	16-Jul-1982
	Include errorlog UCB fields in journal UCB		
V03-005	JSV0019	Joost Verhofstad	12-Jul-1982
	Add UCBSV_KNOWN_JNL		
V03-004	LY0027	Larry Yetto	29-Jun-1982
	Define UCBSL_JNL_NDL to be the same as UCBSL_JNL_ASID		
V03-003	JSV008	Joost Verhofstad	10-Jun-1982
	Add UCB, VCB and WCB fields for journals		
V03-002	KTA0100	Kerbey T. Altmann	07-Jun-1982
	Add field MEDIA_ID to UCB.		
V03-001	KDM0078	Kathleen D. Morse	15-Mar-1982
	Add RPBSV_FINDMEM flag. for 11/782 installations.		

..

```
module $RBMDEF;
```

```
/*  
/* RBM      - realtime bit map of SPTs available for real time processes  
/*-
```

```
aggregate RBMDEF structure prefix RBMS;
```

```
STARTVPN longword unsigned;
```

```
FREECOUNT longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
FILL 1 byte fill prefix RBMDEF tag $$;
```

```
constant 'LENGTH' equals . prefix RBMS tag K;
```

```
constant 'LENGTH' equals . prefix RBMS tag L;
```

```
BITMAP longword unsigned;
```

```
/* Starting VPN of bit map.
```

```
/* Number of free SPTs.
```

```
/* Size of control block.
```

```
/* Type of control block.
```

```
/* Spare byte.
```

```
/* Length of block so far.
```

```
/* Length of block so far.
```

```
/* Start of bit map.
```

```
end RBMDEF;
```

```
end_module $RBMDEF;
```

```
module $RDIDEF;
```

```
/*++
```

```
/* Rights Database Identifier Block definitions. This structure contains the  
/* RMS Internal File Identifiers (IFI's) and Internal Stream Identifiers  
/* (ISI's) for the rights database. This structure is allocated from the  
/* process allocation region pool.
```

```
/*--
```

```
aggregate RDIDEF structure prefix RDIS;
```

```
    #ISI_MAX = 10;
```

```
    constant ISI_MAX equals #ISI_MAX; /* Maximum number of concurrent record streams
```

```
    SIZE longword unsigned; /* Size of allocated block
```

```
    IFI_READ longword unsigned; /* Internal File Identifier for read operations
```

```
    IFI_WRITE longword unsigned; /* Internal File Identifier for write operations
```

```
    ISI_VEC longword unsigned dimension 0:#ISI_MAX; /* Internal Stream Identifier vector
```

```
end RDIDEF;
```

```
end_module $RDIDEF;
```

```
module SRDPDEF;
```

```
/*  
/* REMOTE DEVICE PROTOCOL DEFINITIONS  
/*
```

```
aggregate RDPDEF structure prefix RDP$;
```

```
  OPCODE word unsigned; /*OPERATION CODE  
  MOD word unsigned; /*OPERATION CODE MODIFIERS  
  REFID longword unsigned; /*REFERENCE ID  
  UNIT_OVERLAY union fill;  
    ONIT word unsigned; /*DEVICE UNIT NUMBER  
    constant HEADERLEN equals : prefix RDP$ tag K; /*HEADER LENGTH  
    constant HEADERLEN equals : prefix RDP$ tag C; /*HEADER LENGTH  
    SIZE word unsigned; /*SIZE OF MESSAGE (ACP/DRIVER USE ONLY)  
  end UNIT_OVERLAY;  
  PARAM1 longword unsigned; /*PARAMETER 1  
  PARAM2 longword unsigned; /*PARAMETER 2  
  PARAM3 longword unsigned; /*PARAMETER 3  
  PARAM4 longword unsigned; /*PARAMETER 4  
  PARAM5 longword unsigned; /*PARAMETER 5  
  PARAM6 longword unsigned; /*PARAMETER 6
```

```
/*  
/* RESPONSE FROM REMOTE PACKET DEFINITIONS  
/*
```

```
  constant( /*RESPONSE PACKET OPCODES  
    ATTN /* ATTENTION  
    'END' /* I/O REQUEST COMPLETE  
    LOG /* ERROR LOG  
  ) equals -1 increment -1 prefix RDP tag $C;
```

```
end RDPDEF;
```

```
aggregate RDPDEF1 structure prefix RDP$;
```

```
  FILL 3 byte dimension 10 fill prefix RDPDEF tag $$; /*END PACKET I/O STATUS  
  STATUS quadword unsigned;
```

```
/*  
/* TERMINAL SPECIFIC PARAMETER DEFINITIONS  
/*
```

```
/* READ/WRITE REQUEST
```

```
end RDPDEF1;
```

```
aggregate RDPDEF2 structure prefix RDP$;
```

```
  FILL 4 byte dimension 10 fill prefix RDPDEF tag $$; /*BYTE COUNT  
  TT_BCNT longword unsigned;  
  TT_CARCON OVERLAY union fill; /*WRITE CARRIAGE CONTROL  
    TT_CARCON longword unsigned; /*READ TIMEOUT  
    TT_TIMEOUT longword unsigned;  
  end TT_CARCON_OVERLAY;  
  TT_WDATA OVERLAY union fill;  
    TT_WDATA character; /*WRITE DATA  
    TT_TERM character; /*BYTE OF SIZE + TERMINATOR MASK  
    /*WORD OF SIZE + PROMPT STRING
```

```
/* SET MODE/CHARACTERISTICS REQUEST
```

```
  end TT_WDATA_OVERLAY;
```

```
end RDPDEF2;
```

```

aggregate RDPDEF3 structure prefix RDP$;
  FILL_5 byte dimension 10 fill prefix RDPDEF tag $$;
  TT_CHAR_OVERLAY union fill;
    TT_CHAR quadword unsigned; /*CHARACTERISTICS
    TT_ASTPRM longword unsigned; /*AST PARAMETER
  end TT_CHAR_OVERLAY;
  TT_SPEED longword unsigned; /*LINE SPEED
  TT_FILL longword unsigned; /*FILL SPECIFIER
  TT_PARITY longword unsigned; /*PARITY FLAGS
  TT_CHAR2 longword unsigned; /* Remaining longword of characters
/* READ REQUEST END PACKET
end RDPDEF3;

aggregate RDPDEF4 structure prefix RDP$;
  FILL_6 byte dimension 10 fill prefix RDPDEF tag $$;
  FILL_1 quadword fill prefix RDPDEF tag $$; /*I/O STATUS
  TT_RDATA character; /*WORD OF SIZE + READ DATA
/* SENSE MODE/CHARACTERISTICS END PACKET
end RDPDEF4;

aggregate RDPDEF5 structure prefix RDP$;
  FILL_7 byte dimension 10 fill prefix RDPDEF tag $$;
  FILL_2 quadword fill prefix RDPDEF tag $$; /*I/O STATUS
  TT_SCHAR quadword unsigned; /*SENSED CHARACTERISTICS
  TT_SCHAR2 longword unsigned; /* Additional longword of characters
/* Broadcast message attention packet
end RDPDEF5;

aggregate RDPDEF6 structure prefix RDP$;
  FILL_8 byte dimension 10 fill prefix RDPDEF tag $$;
  TT_BRDTOTSIZE word unsigned; /* Total size of data
  TT_BRDMSG word unsigned; /* Message code
  TT_BRDUNIT word unsigned; /* Unit number
  TT_BRDNAME character length 16; /* Device name as counted string
  constant TT_BRDNAME equals 16 prefix RDP tag $C; /* Size of name field
  TT_BRDXTSIZE_OVERLAY union fill;
    TT_BRDXTSIZE word unsigned; /* Count for message text
    TT_BRDXTSIZE_FIELDS structure fill;
      FILL_9 byte dimension 2 fill prefix RDPDEF tag $$;
      TT_BRDTEXT character length 0 tag T; /* Message text start
/* Out of band attention packet
  end TT_BRDXTSIZE_FIELDS;
  end TT_BRDXTSIZE_OVERLAY;
end RDPDEF6;

aggregate RDPDEF7 structure prefix RDP$;
  FILL_10 byte dimension 10 fill prefix RDPDEF tag $$;
  TT_OOTBAND byte unsigned; /* Out of band character
/* ATTENTION PACKET MODIFIERS
  constant(
    TT_UNSQL /*UNSOLICITED DATA

```

```
, TT_HANGUP
, TT_CTRLC
, TT_CTRLY
, TT_STARTRCV
, TT_BRDCST
, TT_OUTBAND
) equals 0 increment 1 prefix RDP tag $C;
```

end RDPDEF7;

end_module \$RDPDEF;

```
/*MODEM HANGUP
/*CONTROL/C
/*CONTROL/Y
/* Start a receive to the net
/* Broadcast message for mailbox
/* Out of band AST
```

```
module $RBFDEF;
```

```
/*
/*      Remote buffer as stored in dynamic memory
/*
/*      This structure must be identical to the above structure except
/*      for the header, which is the header for a buffered io buffer.
/*
```

```
/*
/*      Buffered io buffer header
/*
```

```
aggregate RBFDEF structure prefix RBF$;
```

```
MSGDAT longword unsigned; /* Address of message data
USRBFR longword unsigned; /* User buffer address
SIZE word unsigned; /* Size of structure
TYPE byte unsigned; /* Type of structure, DYN$C_BUFIO
SPARE byte unsigned; /* Alignment
DATSIZE word unsigned; /* Data size
```

```
/*
/*      End of header
/*
```

```
OPCODE word unsigned; /*OPERATION CODE
MOD word unsigned; /*OPERATION CODE MODIFIERS
REFID longword unsigned; /*REFERENCE ID
UNIT word unsigned; /*DEVICE UNIT NUMBER
/*      S      SIZE,0,W /*SIZE OF MESSAGE (ACP/DRIVER USE ONLY)
constant HEADERLEN equals . prefix RBF$ tag K; /*HEADER LENGTH
constant HEADERLEN equals . prefix RBF$ tag C; /*HEADER LENGTH
PARAM1 longword unsigned; /*PARAMETER 1
PARAM2 longword unsigned; /*PARAMETER 2
PARAM3 longword unsigned; /*PARAMETER 3
PARAM4 longword unsigned; /*PARAMETER 4
PARAM5 longword unsigned; /*PARAMETER 5
PARAM6 longword unsigned; /*PARAMETER 6
```

```
/*
/* RESPONSE FROM REMOTE PACKET DEFINITIONS
/*
```

```
constant( /*RESPONSE PACKET OPCODES
    ATTN /* ATTENTION
    'END' /* I/O REQUEST COMPLETE
    LOG /* ERROR LOG
) equals -1 increment -1 prefix RBF tag $C;
end RBFDEF;
```

```
aggregate RBFDEF1 structure prefix RBF$;
```

```
FILL 3 byte dimension 24 fill prefix RBFDEF tag $S; /*END PACKET I/O STATUS
STATOS quadword unsigned;
```

```
/*
/* TERMINAL SPECIFIC PARAMETER DEFINITIONS
/*
```

```
/* READ/WRITE REQUEST
end RBFDEF1;
```

```
aggregate RBFDEF2 structure prefix RBF$;
  FILL_4 byte dimension 24 fill prefix RBFDEF tag $$;
  TT_BCNT longword unsigned; /*BYTE COUNT
  TT_CARCON OVERLAY union fill;
    TT_CARCON longword unsigned; /*WRITE CARRIAGE CONTROL
    TT_TIMEOUT longword unsigned; /*READ TIMEOUT
  end TT_CARCON OVERLAY;
  TT_WDATA OVERLAY union fill;
    TT_WDATA character; /*WRITE DATA
    TT_TERM character; /*BYTE OF SIZE + TERMINATOR MASK
    /*WORD OF SIZE + PROMPT STRING
```

```
/* SET MODE/CHARACTERISTICS REQUEST
  end TT_WDATA_OVERLAY;
end RBFDEF2;
```

```
aggregate RBFDEF3 structure prefix RBF$;
  FILL_5 byte dimension 24 fill prefix RBFDEF tag $$;
  TT_CHAR OVERLAY union fill;
    TT_CHAR quadword unsigned; /*CHARACTERISTICS
    TT_ASTPRM longword unsigned; /*AST PARAMETER
  end TT_CHAR OVERLAY;
  TT_SPEED longword unsigned; /*LINE SPEED
  TT_FILL longword unsigned; /*FILL SPECIFIER
  TT_PARITY longword unsigned; /*PARITY FLAGS
  TT_CHAR2 longword unsigned; /* Another longword of characters
/* READ REQUEST END PACKET
end RBFDEF3;
```

```
aggregate RBFDEF4 structure prefix RBF$;
  FILL_6 byte dimension 24 fill prefix RBFDEF tag $$;
  FILL_1 quadword fill prefix RBFDEF tag $$; /*I/O STATUS
  TT_RDATA character; /*WORD OF SIZE + READ DATA
/* SENSE MODE/CHARACTERISTICS END PACKET
end RBFDEF4;
```

```
aggregate RBFDEF5 structure prefix RBF$;
  FILL_7 byte dimension 24 fill prefix RBFDEF tag $$;
  FILL_2 quadword fill prefix RBFDEF tag $$; /*I/O STATUS
  TT_SCHAR quadword unsigned; /*SENSED CHARACTERISTICS
  TT_SCHAR2 longword unsigned; /* Another longword of characters
```

```
/* Broadcast message attention packet
end RBFDEF5;
```

```
aggregate RBFDEF6 structure prefix RBF$;
  FILL_8 byte dimension 24 fill prefix RBFDEF tag $$;
  TT_BRDTOTSIZE word unsigned; /* Total size of data
  TT_BRDMSG word unsigned; /* Message code
  TT_BRDUNIT word unsigned; /* Unit number
  TT_BRDNAME character length 16; /* Device name as counted string
  constant TT_BRDNAME equals 16 prefix RBF tag $C; /* Size of name field
  TT_BRDTEXTSIZE OVERLAY union fill;
    TT_BRDTEXTSIZE word unsigned; /* Count for message text
```

```
TT_BRDTXTSIZE_FIELDS structure fill;
  FILL 9 byte dimension 2 fill prefix RBFDEF tag $$;
  TT_BRDTEXT character length 0 tag T; /* Message text start
/* Out of band attention packet
  end TT_BRDTXTSIZE_FIELDS;
end TT_BRDTXTSIZE_OVERLAY;
end RBFDEF5;
aggregate RBFDEF7 structure prefix RBFS;
  FILL 10 byte dimension 24 fill prefix RBFDEF tag $$;
  TT_OUTBAND byte unsigned; /* Out of band character
/* ATTENTION PACKET MODIFIERS
  constant(
    TT_UN SOL /*UNSOLICITED DATA
    , TT_HANGUP /*MODEM HANGUP
    , TT_CTRL C /*CONTROL/C
    , TT_CTRL Y /*CONTROL/Y
    , TT_STARTRCV /* Start a receive to the net
    , TT_BRDCST /* Broadcast message for mailbox
    , TT_OUTBAND /* Out of band AST
  ) equals 0 increment 1 prefix RBF tag $C;
end RBFDEF7;
end_module $RBFDEF;
```

```
module $RCTDEF;
```

```
/*+
/* RCT - Replacement and Caching Table sector !0 layout.
/* The RCT is a structure residing on disks controlled by MSCP
/* speaking disk controllers. The RCT is maintained by the intelligent
/* controllers and the disk class driver. The disk class driver mainly
/* gets involved in RCT manipulations during host initiated bad
/* block replacement.
```

```
aggregate RCTDEF structure prefix RCT$;
```

```
  VOLSER quadword unsigned; /* Volume serial number
  FLAGS OVERLAY union fill; /* Flags word
    FLAGS word unsigned; /* Write back caching in use
    FLAGS BITS structure fill; tag $$;
      WB bitfield mask; /* Forced Error flag for block being replaced
      FILL 1 bitfield length 6 fill prefix RCTDEF tag $$;
      FE bitfield mask; /* Replacement caused by Bad RBN
      FILL 2 bitfield length 5 fill prefix RCTDEF tag $$;
      BR bitfield mask; /* Replacement in Progress phase 2
      RP2 bitfield mask; /* Replacement in Progress phase 1
      RP1 bitfield mask;
    end FLAGS BITS;
  end FLAGS OVERLAY;
  FILL 3 word fill prefix RCTDEF tag $$; /* Reserved word
  LBN longword unsigned; /* LBN currently being replaced.
  RBN longword unsigned; /* RBN allocated to replace LBN
  BAD_RBN longword unsigned; /* If BR flag, RBN of bad replacement block
  WB_CTRL quadword unsigned; /* Serial ! of last controller doing Write back
  WB_INCAR longword unsigned; /* Write back incarnation !
  INCARTIME OVERLAY union fill;
    INCARTIME quadword unsigned; /* Date-time of last update of incarnation no.
```

```
/*
/* Structure of a Replacement Block Descriptor
/*
```

```
  INCARTIME BITS0 structure fill;
    LBN bitfield mask length 28; /* Space for LBN replaced by this RBN
    CODE bitfield mask length 4; /* Describes how this descriptor being used
  end INCARTIME_BITS0;

  INCARTIME BITS1 structure fill;
    FILL 4 bitfield length 28 fill prefix RCTDEF tag $$; /* LBN
    NONPRIME bitfield mask; /* Set implies allocated, but not prime RBN
    ALLOCATED bitfield mask; /* This RBN allocated
    UNUSABLE bitfield mask; /* This RBN unusable
    NULL bitfield mask; /* This marks a NULL entry
  end INCARTIME_BITS1;

  /* Values of CODE
  constant EMPTY equals 0 prefix RCT tag $K; /* Unallocated (empty) replacement block
  constant ALOCPRIME equals 2 prefix RCT tag $K; /* Allocated replace blk - primary RBN
  constant ALOCNONP equals 3 prefix RCT tag $K; /* Allocated replace blk - non-primary RBN
  constant UNUSABLE equals 4 prefix RCT tag $K; /* Unusable replacement block
  constant ALTUNUSE equals 5 prefix RCT tag $K; /* Alternate unusable replacement block
  constant NULL equals 8 prefix RCT tag $K; /* Null entry - no corresponding RBN sector
end INCARTIME_OVERLAY;
```

end RCTDEF;

end_module \$RCTDEF;

module \$RDTDEF;

/*

/* RDT - SCS RESPONSE DESCRIPTOR TABLE

/*

/* ONE RESPONSE DESCRIPTOR (RD) IS ALLOCATED FOR EACH SCS MESSAGE

/* SENT FOR WHICH THE SENDER EXPECTS A MATCHING RESPONSE.

/*-

aggregate \$RDTDEF structure prefix RDT\$ origin FILL_2;

WAITFL longword unsigned;

WAITBL longword unsigned;

SIZE word unsigned;

TYPE byte unsigned;

SUBTYP byte unsigned;

FREERD longword unsigned;

MAXRDIDX longword unsigned;

FILL 1 longword fill prefix RDTDEF tag \$\$;

constant 'LENGTH' equals 24 prefix RDT tag \$C;

/*

FILL 2 byte fill prefix RDTDEF tag \$\$;

end RDTDEF;

end_module \$RDTDEF;

/*RD WAIT QUEUE FWD LINK

/*RD WAIT QUEUE BACK LINK

/*STRUCTURE SIZE IN BYTES

/*SCS STURCTURE TYPE

/*SCS STRUCT SUBTYPE FOR RDT

/*ADDR OF 1ST FREE RD

/*MAXIMUM ! OF DESCRIPTORS

/*RESERVED FOR FUTURE USE

/*LENGTH OF NEG PORTION OF STRUCTURE

```
module $RDDEF;
```

```
/*
```

```
/* RD - SCS RESPONSE DESCRIPTOR FORMAT
```

```
*/
```

```
aggregate RDDEF structure prefix RD$;
```

```
  CDRP OVERLAY union fill;
```

```
    CDRP longword unsigned;
```

```
    LINK longword unsigned;
```

```
  end CDRP OVERLAY;
```

```
  STATE OVERLAY union fill;
```

```
    STATE word unsigned;
```

```
    STATE BITS structure fill;
```

```
      BSY bitfield;
```

```
      PERM bitfield;
```

```
    end STATE BITS;
```

```
  end STATE OVERLAY;
```

```
  SEQNUM word unsigned;
```

```
  constant 'LENGTH' equals . prefix RD$ tag K;
```

```
  constant 'LENGTH' equals . prefix RD$ tag C;
```

```
/*ADDR OF ASSOC CDRP OR  
/* OR OTHER CONTEXT BLOCK  
/* OR LINK TO NEXT FREE RD
```

```
/*RD STATE FLAGS
```

```
/* ALLOCATED IF SET
```

```
/* PERMANENTLY ALLOCATED RD IF SET
```

```
/*SEQUENCE NUMBER OF RD
```

```
/*LENGTH OF RD
```

```
/*LENGTH OF RD
```

```
end RDDEF;
```

```
end_module $RDDEF;
```

```
end
```

```
/*
```

```
/*
```

```
/*
```

```
agg
```

```
end
```

```
end
```



```
module SRPBDEF;
```

```
/*  
/* RESTART PARAMETER BLOCK DEFINITIONS  
/*
```

```
aggregate RPBDEF structure prefix RPB$;
```

```
BASE longword unsigned;  
RESTART longword unsigned;  
CHKSUM longword unsigned;  
RSTRFLG longword unsigned;  
HALTPC longword unsigned;  
HALTPSL longword unsigned;  
HALTCODE longword unsigned;  
BOOTRO OVERLAY union fill;
```

```
    BOOTRO longword unsigned;  
    BOOTRO_FIELDS structure fill;
```

```
        RODEVTYP byte unsigned;  
        FILL 1 byte fill prefix RPBDEF tag $$;  
        ROUBVEC word unsigned;
```

```
    end BOOTRO_FIELDS;
```

```
end BOOTRO_OVERLAY;
```

```
BOOTR1 OVERLAY union fill;
```

```
    BOOTR1 longword unsigned;  
    BOOTR1_BITS structure fill;  
        NEXUS bitfield length 4;  
        ABUS bitfield length 2;
```

```
    end BOOTR1_BITS;
```

```
end BOOTR1_OVERLAY;
```

```
BOOTR2 longword unsigned;
```

```
BOOTR3 longword unsigned;
```

```
BOOTR4 longword unsigned;
```

```
BOOTR5 OVERLAY union fill;
```

```
    BOOTR5 longword unsigned;  
    BOOTR5_BITS structure fill;
```

```
        CONV bitfield;
```

```
        DEBUG bitfield;
```

```
        INIBPT bitfield;
```

```
        BBLOCK bitfield;
```

```
        DIAG bitfield;
```

```
        BOOBPT bitfield;
```

```
        HEADER bitfield;
```

```
        NOTEST bitfield;
```

```
        SOLICT bitfield;
```

```
        HALT bitfield;
```

```
        NOPFND bitfield;
```

```
        MPM bitfield mask;
```

```
        USEMPM bitfield mask;
```

```
        MEMTEST bitfield mask;
```

```
        FINDMEM bitfield mask;
```

```
        AUTOTEST bitfield mask;
```

```
        CRDTEST bitfield mask;
```

```
        FILL 2 bitfield length 11 fill prefix RPBDEF tag $;
```

```
        TOPSYS bitfield mask length 4;
```

```
    end BOOTR5_BITS;
```

```
end BOOTR5_OVERLAY;
```

```
/*PHYSICAL BASE ADDRESS OF 64K BLOCK  
/*POINTER TO RESTART ROUTINE (PHYSICAL)  
/*CHECKSUM OF BYTES 0-7F OF RESTART ROUTINE  
/*RESTART IN PROGRESS FLAG  
/*PC AT RESTART/HALT  
/*PSL AT RESTART/HALT  
/*CODE DESCRIBING RESTART REASON
```

```
/*SAVED BOOT PARAMETER R0
```

```
/* DEVICE TYPE SUBFIELD
```

```
/* RESERVED
```

```
/* UNIBUS INT VECTOR SUBFIELD
```

```
/*SAVED BOOT PARAMETER R1
```

```
/*NEXUS OF SYSTEM DEVICE ADAPTER
```

```
/*ABUS ADAPTER NUMBER OF SBIA
```

```
/*SAVED BOOT PARAMETER R2
```

```
/*SAVED BOOT PARAMETER R3
```

```
/*SAVED BOOT PARAMETER R4
```

```
/*SAVED BOOT PARAMETER R5
```

```
/* CONVERSATIONAL BOOTSTRAP
```

```
/* KEEP DEBUGGER CODE
```

```
/* INITIAL BREAKPOINT
```

```
/* TRANSFER TO BOOTBLOCK
```

```
/* BOOT DIAGNOSTIC FILE
```

```
/* BOOTSTRAP BREAKPOINT
```

```
/* USE START ADDRESS FROM IMAGE HEADER
```

```
/* FLAG TO INHIBIT MEMORY TESTING
```

```
/* SOLICIT BOOT FILE NAME
```

```
/* HALT BEFORE TRANSFER
```

```
/* INHIBIT PFN DELETION
```

```
/* MULTI-PROCESSOR BOOT, USE MA780 ONLY
```

```
/* USE MA780 AS IF IT WERE LOCAL MEMORY
```

```
/* USE STRICTER TEST TO VALIDATE MEMORY
```

```
/* FIND SUFFICIENT MEMORY TO BOOT (>512K)
```

```
/* USED BY DIAGNOSTIC SUPERVISOR
```

```
/* REMOVE PAGES WITH CRD ERRORS
```

```
/*SYSTEM DIRECTORY NUMBER
```

SYS

mod

/*

/*

/*

/*

/*

/*

agg

end

end

```

IOVEC longword unsigned; /*ADDRESS OF BOOTSTRAP QIO VECTOR
IOVECSZ longword unsigned; /*SIZE OF BOOT QIO ROUTINE
FILLBN longword unsigned; /*LOGICAL BLOCK NUMBER OF BOOT FILE
FILSIZ longword unsigned; /*SIZE OF BOOT FILE
PFNMAP quadword unsigned; /*DESCRIPTOR FOR PFN BITMAP
PFNCNT longword unsigned; /*COUNT OF PHYSICAL PAGES
SVASPT longword unsigned; /*SYSTEM VIRTUAL ADDRESS OF SPT
CSRPHY longword unsigned; /*UBA DEVICE CSR ADDRESS (PHYSICAL)
CSRVR longword unsigned; /*UBA DEVICE CSR ADDRESS (VIRTUAL)
ADPPHY longword unsigned; /*ADAPTER CONFIGURATION REGISTER (PHYSICAL)
ADPVIR longword unsigned; /*ADAPTER CONFIGURATION REGISTER (VIRTUAL)
UNIT word unsigned; /*UNIT NUMBER
DEVTYP byte unsigned; /*DEVICE TYPE CODE
SLAVE byte unsigned; /*SLAVE UNIT NUMBER
FILE character length 40; /*BOOT FILE NAME (ASCII)
CONFREG byte unsigned dimension 16; /*ARRAY OF ADAPTER TYPES
HDRPGCNT byte unsigned; /*COUNT OF HEADER PAGES
BOOTNDT OVERLAY union fill;
    BOOTNDT word unsigned; /*16-BIT BOOT ADAPTER NEXUS DEVICE TYPE
    BOOTNDT byte unsigned; /* 8-BIT BCOT ADAPTER NEXUS DEVICE TYPE
end BOOTNDT OVERLAY;
FLAGS structure byte unsigned; /*MISCELLANEOUS FLAG BITS
    NOSYSDISK bitfield mask; /* BOOT DISK IS NOT PRESENT
end FLAGS;
ISP longword unsigned; /*PWR FAIL INTERRUPT STACK POINTER
PCBB longword unsigned; /*PROCESS CONTROL BLOCK BASE
SBR longword unsigned; /*SYSTEM BASE REGISTER
SCBB longword unsigned; /*SYSTEM CONTROL BLOCK BASE
SISR longword unsigned; /*SOFTWARE INTERRUPT SUMMARY REGISTER
SLR longword unsigned; /*SYSTEM LENGTH REGISTER
MEMDSC OVERLAY union fill;
    MEMDSC longword unsigned dimension 16; /*MEMORY DESCRIPT. - PAGCNT, TR, BASE PFN
    MEMDSC BITS structure fill;
        PAGCNT bitfield length 24; /* COUNT OF PAGES FOR THIS MEMORY
        TR bitfield length 8; /* TR NUMBER FOR THIS MEMORY
        BASEPFN bitfield length 32; /* BASE PFN FOR THIS MEMORY
    end MEMDSC BITS;
    constant MEMDSCSIZ equals 8 prefix RPB tag $C; /*NUMBER OF BYTES IN ONE MEM DESCRIPTOR
    constant NMEMDSC equals 8 prefix RPB tag $C; /*NUMBER OF MEMORY DESCRIPTORS IN RPB
end MEMDSC OVERLAY;
BUGCHK longword unsigned; /*BUGCHECK LOOP ADDRESS FOR MP SECONDARY
WAIT byte unsigned dimension 4; /*BUGCHECK LOOP CODE FOR MP SECONDARY
BADPGS longword unsigned; /*NUMBER OF BAD PAGES FOUND IN MEM SCAN
CTRLTR byte unsigned; /*CONTROLLER LETTER DESIGNATOR
constant 'LENGTH' equals . prefix RPB$ tag K; /*LENGTH OF RPB
constant 'LENGTH' equals . prefix RPB$ tag C; /*LENGTH OF RPB
end RPBDEF;

end_module $RPBDEF;

```

```
module $RSBDEF;
```

```
/*  
/* RSB - RESOURCE BLOCK  
/*  
/* RESOURCE BLOCKS REPRESENT RESOURCES FOR WHICH THERE ARE LOCKS OUTSTANDING.  
/* EACH RESOURCE BLOCK MAY HAVE ONE OR MORE LOCK BLOCKS (LKB) QUEUED TO IT.  
/*-
```

```
aggregate RSBDEF structure prefix RSB$;
```

```
  HSHCHN longword unsigned; /*HASH CHAIN  
  HSHCHNBK longword unsigned; /*HASH CHAIN BACK POINTER  
  SIZE word unsigned; /*SIZE OF RSB  
  TYPE byte unsigned; /*STRUCTURE TYPE  
  DEPTH byte unsigned; /*DEPTH IN TREE  
  GGMODE byte unsigned; /*GROUP GRANT MODE  
  CGMODE byte unsigned; /*CONVERSION GRANT MODE  
  STATUS OVERLAY union fill;  
    STATUS word unsigned; /*STATUS  
    STATUS BITS structure fill;  
      DIRENTRY bitfield mask; /* ENTERED IN DIR. DURING FAILOVER  
      VALINVLD bitfield mask; /* VALUE BLOCK INVALID  
    end STATUS BITS;  
  end STATUS OVERLAY;  
  GRQFL longword unsigned; /*GRANTED QUEUE FORWARD LINK  
  GRQBL longword unsigned; /*GRANTED QUEUE BACKWARD LINK  
  CVTQFL longword unsigned; /*CONVERSION QUEUE FORWARD LINK  
  CVTQBL longword unsigned; /*CONVERSION QUEUE BACKWARD LINK  
  WTQFL longword unsigned; /*WAIT QUEUE FORWARD LINK  
  WTQBL longword unsigned; /*WAIT QUEUE BACKWARD LINK  
  VALBLK quadword unsigned; /*VALUE BLOCK  
  FILL_1 quadword fill prefix RSBDEF tag $$; /*MORE VALUE BLOCK  
  CSID longword unsigned; /*SYSTEM ID OF MASTER SYS.  
  VALSEQNUM longword unsigned; /*VALUE BLOCK SEQ. NUMBER  
  REFCNT word unsigned; /*SUB RSB REFERENCE COUNT  
  BLKASTCNT word unsigned; /*BLOCKING AST COUNT  
  HASHVAL word unsigned; /*HASH VALUE  
  RQSEQNM word unsigned; /*REQUEST SEQUENCE NUMBER  
  PARENT longword unsigned; /*ADDRESS OF PARENT RSB  
  GROUP word unsigned; /*GROUP NUMBER  
  RMOD byte unsigned; /*ACCESS MODE OF RESOURCE  
  RSNLEN byte unsigned; /*RESOURCE NAME LENGTH  
  constant 'LENGTH' equals . prefix RSB$ tag K; /*LENGTH OF FIXED PART OF RSB  
  constant 'LENGTH' equals . prefix RSB$ tag C; /*LENGTH OF FIXED PART OF RSB  
  RESNAM character length 0 tag I; /*START OF RESOURCE NAME  
  constant MAXLEN equals 31 prefix RSB tag $K; /*MAXIMUM LENGTH OF RESOURCE NAME
```

```
end RSBDEF;
```

```
end_module $RSBDEF;
```

module \$RSNDEF;

/*
/* RESOURCE NAME DEFINITIONS
/*

constant(

ASTWAIT
MAILBOX
NPDYNMEM
PGFILE
PGDYNMEM
BRKTHRU
IACLOCK
JQUOTA
LOCKID
SWPFILE
MPLEMPY
MPWBUSY
SCS
CLUSTAN
MAX
equals 1 increment 1 prefix RSN tag \$;

end_module \$RSNDEF;

/*0 ORIGIN IN INCREMENTS OF 1

/*WAIT FOR AST EVENT, CHANNEL INTERLOCK
/*MAILBOX SPACE
/*NON-PAGED DYNAMIC MEMORY
/*PAGING FILE SPACE
/*PAGED DYNAMIC MEMORY
/*TERMINAL BROADCAST
/*IMAGE ACTIVATION INTERLOCK
/*JOB POOLED QUOTA
/*LOCKIDS
/*SWAPPING FILE SPACE
/*MODIFIED PAGE LIST EMPTY
/*MODIFIED PAGE WRITER BUSY
/*SYSTEM COMMUNICATION
/*CLUSTER STATE TRANSITION
/*MAXIMUM RESOURCE NUMBER

```
module $RUCBDEF;
```

```
/*
```

```
/* Internal control block for use by Recovery Unit services.
```

```
/*
```

```
/* This structure is created when a process first issues a
```

```
/* RUF service and is pointed to by CTL$$_RUF.
```

```
/*-
```

```
aggregate RUCBDEF structure fill prefix RUCBS;
```

```
  RUID OVERLAY union fill;
```

```
    RUID character length 16;
```

```
    constant ID_LEN equals . prefix RUCBS tag K; /* 128-bit id
```

```
    constant ID_LEN equals . prefix RUCBS tag C; /* length of the ID
```

```
    RUID FIELDS structure fill;
```

```
      ID_TIME quadword unsigned;
```

```
      /* system time in 100ns. units
```

```
      ID_CSID longword unsigned;
```

```
      /* cluster ID
```

```
      ID_EPID longword unsigned;
```

```
      /* cluster PID of initiating process
```

```
    end RUID FIELDS;
```

```
  end RUID OVERLAY;
```

```
  RU_CTRL OVERLAY union fill;
```

```
    RU_CTRL longword unsigned;
```

```
    /* control longword
```

```
    RU_CTRL FIELDS structure fill;
```

```
      STATE byte unsigned;
```

```
      /* RU current state
```

```
      constant ACTIVE equals 1 prefix RUCB tag $C; /* active
```

```
      constant PH1_INIT equals 2 prefix RUCB tag $C; /* phase 1 commit started
```

```
      constant PH1_FIN equals 3 prefix RUCB tag $C; /* phase 1 commit completed
```

```
      constant PH2_INIT equals 4 prefix RUCB tag $C; /* phase 2 commit started
```

```
      constant PH2_FIN equals 5 prefix RUCB tag $C; /* phase 2 commit completed
```

```
      constant CANCEL equals 6 prefix RUCB tag $C; /* cancel in progress
```

```
      constant RESET equals 7 prefix RUCB tag $C; /* reset to markpoint in progress
```

```
    CTRL OVERLAY union fill;
```

```
      CTRL byte unsigned;
```

```
      /* control flags
```

```
      CTRL BITS structure fill;
```

```
        INIT bitfield mask;
```

```
        /* RUCB has been initialized
```

```
        ACTIVE bitfield mask;
```

```
        /* Recovery Unit is currently active
```

```
        INHANDLER bitfield mask;
```

```
        /* processing handlers
```

```
        RUSYNC bitfield mask;
```

```
        /* handler has specified synchronous RUs
```

```
      end CTRL BITS;
```

```
    end CTRL OVERLAY;
```

```
    SRVCODE word unsigned;
```

```
    /* current operation change mode code
```

```
  end RU_CTRL FIELDS;
```

```
end RU_CTRL OVERLAY;
```

```
HACTION longword unsigned;
```

```
/* handler action code
```

```
HREASON longword unsigned;
```

```
/* handler invocation reason code (see $RUHRSNDEF)
```

```
MARKPT longword unsigned;
```

```
/* most recent markpoint value
```

```
RUH_EXEC longword unsigned;
```

```
/* exec mode handler listhead
```

```
RUH_SUPV longword unsigned;
```

```
/* supervisor mode handler listhead
```

```
RUH_USER longword unsigned;
```

```
/* user mode handler listhead
```

```
SV_PCPSL quadword unsigned;
```

```
/* saved user's PC and PSL from original CHMK
```

```
SV_RSB longword unsigned;
```

```
/* saved return address from RUF$CALL_HANDLERS
```

```
FREESP longword unsigned;
```

```
/* free handler cell listhead
```

```
AILIST longword unsigned;
```

```
/* ptr to list of AI jnls touched in RU
```

```
BILIST longword unsigned;
```

```
/* ptr to list of BI jnls touched in RU
```

```
IMPURE longword unsigned;
```

```
/* ptr to impure storage area for ENDRU
```

```
AMODE byte unsigned;
```

```
/* mode of caller of RUF$START
```

```
SV_AMODE byte unsigned;
```

```
/* saved access mode (R4) from RUF$CALL_HANDLERS
```

```
FILL 1 byte dimension 2 fill prefix RUCBDEF tag SS; /* spare
constant 'LENGTH' equals . prefix RUCB$ tag K; /* length of RUCB
constant 'LENGTH' equals . prefix RUCB$ tag C; /* length of RUCB
end RUCBDEF;
end_module $RUCBDEF;
```

module \$RUHDEF;

/*

/* Recovery Unit Handler storage cell definitions

/*

aggregate RUHDEF structure fill prefix RUH\$;

LINK longword unsigned;

ADDR longword unsigned;

PARAM longword unsigned;

constant SIZE equals . prefix RUH\$ tag K;

constant SIZE equals . prefix RUH\$ tag C;

end RUHDEF;

end_module \$RUHDEF;

/* Next cell address ptr

/* Routine addr

/* Routine parameter

/* Size of cell

/* Size of cell

mod

/*

/*

/*

/*

/*

/*

/*

/*

agg

con

end

/*

/*

/*

/*

/*

/*

/*

con

con

con

con

enc

```
module $RVTDEF;
```

```
/*  
/* RVT - RELATIVE VOLUME TABLE  
/*  
/* A RELATIVE VOLUME MAPPING TABLE IS REQUIRED FOR EVERY MULTIVOLUME  
/* STRUCTURE THAT IS MOUNTED IN A SYSTEM.  
/*-
```

```
aggregate RVTDEF structure prefix RVT$;
```

```
  STRUCLKID longword unsigned; /* LOCK ID OF VOLUME SET LOCK.  
  REFC word unsigned; /* REFERENCE COUNT  
  ACTIVITY word unsigned; /* ACTIVITY COUNT/FLAG  
  SIZE word unsigned; /* SIZE OF RVT IN BYTES  
  TYPE byte unsigned; /* STRUCTURE TYPE OF RVT  
  NVOLS byte unsigned; /* NUMBER OF VOLUMES IN SET  
  STRUCNAME character length 12; /* STRUCTURE (VOLUME SET) NAME  
  VLSLCKNAM character length 12; /* Volume set lock name.  
  BLOCKID longword unsigned; /* Blocking lock id.  
  ACB byte unsigned dimension 28; /* ACB for blocking ast.  
  constant 'LENGTH' equals . prefix RVT$ tag K; /* LENGTH OF STANDARD RVT  
  constant 'LENGTH' equals . prefix RVT$ tag C; /* LENGTH OF STANDARD RVT  
  UCBLST longword unsigned; /* ADDRESSES OF THE RESPECTIVE UCB'S  
  constant MINSIZE equals 18 prefix RVT tag $C; /* MINIMUM NUMBER OF ENTRIES TO ALLOCATE
```

```
end RVTDEF;
```

```
end_module $RVTDEF;
```

```
module $SBDEF;
```

```
/*
```

```
/* SB - SCS SYSTEM BLOCK
```

```
/*
```

```
/* THE SB HAS INFORMATION ABOUT KNOWN SYSTEMS IN A CPU CLUSTER.
```

```
/*-
```

```
aggregate SBDEF structure prefix SB$;
```

```
FLINK longword unsigned;
```

```
BLINK longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYP byte unsigned;
```

```
PBFL longword unsigned;
```

```
PBBL longword unsigned;
```

```
PBCONN longword unsigned;
```

```
SYSTEMID byte unsigned dimension 6;
```

```
FILL 1 word fill prefix SBDEF tag $$;
```

```
MAXDG word unsigned;
```

```
MAXMSG word unsigned;
```

```
SWTYPE character length 4;
```

```
SWVERS character length 4;
```

```
SWINCARN quadword unsigned;
```

```
HWTYPE character length 4;
```

```
HWVERS byte unsigned dimension 12;
```

```
NODENAME character length 16;
```

```
DDB longword unsigned;
```

```
TIMEOUT word;
```

```
ENBMSK byte unsigned dimension 2;
```

```
CSB longword unsigned;
```

```
constant 'LENGTH' equals . prefix SB$ tag K;
```

```
constant 'LENGTH' equals . prefix SB$ tag C;
```

```
/*FWD LINK TO NEXT SB
```

```
/*BACK LINK TO PREVIOUS SB
```

```
/*STRUCTURE SIZE IN BYTES
```

```
/*SCS STRUCTURE TYPE
```

```
/*SCS STRUCT SUBTYPE FOR SB
```

```
/*LINK TO NEXT PATH BLOCK
```

```
/*LINK TO PREVIOUS PATH BLOCK
```

```
/*ADDR OF NEXT PB TO USE FOR
```

```
/* A CONNECTION
```

```
/*SYSTEM ID
```

```
/*RESERVED WORD
```

```
/*MAXIMUM DATAGRAM SIZE
```

```
/*MAXIMUM MESSAGE SIZE
```

```
/*SOFTWARE TYPE, 1-4 CHAR
```

```
/*SOFTWARE VERSION, 1-4 CHAR
```

```
/*SOFTWARE INCARNATION #
```

```
/*HW TYPE, 1-4 CHAR, BLANK FILL
```

```
/*HW VERSION #
```

```
/*SCS NODENAME, COUNTED ASCII STRING
```

```
/*DDB LIST HEAD
```

```
/*SCA PROCESS POLLER, WAITING TIME REMAINING
```

```
/*SCA PROCESS POLLER, PROCESS ENABLE MASK
```

```
/*LINK TO NEWEST CLUSTER SYSTEM BLOCK
```

```
/*LENGTH OF SB
```

```
/*LENGTH OF SB
```

```
end SBDEF;
```

```
end_module $SBDEF;
```

```
module $SBODEF;
```

```
/*  
/* SBO - SCS CONFIG_SYS OUTPUT ARRAY FORMAT  
/*  
/* THE OUTPUT ARRAY RETURNED FROM CALL TO SCSSCONFIG_SYS. DATA IS MOSTLY COPIED FROM  
/* THE SYSTEM BLOCK (SB) BEING LOOKED UP.  
/*-
```

```
aggregate SBODEF structure prefix SBO$;
```

```
SYSTEMID byte unsigned dimension 6;  
FILL 1 word fill prefix SBODEF tag $$;  
MAXDG word unsigned;  
MAXMSG word unsigned;  
SWTYPE character length 4;  
SWVERS character length 4;  
SWINCARN quadword unsigned;  
HWTYPE character length 4;  
HWVERS byte unsigned dimension 12;  
NODENAME character length 16;  
constant VC1 equals . prefix SBO$ tag C;  
constant VC1 equals . prefix SBO$ tag K;  
RSTATION1 byte unsigned dimension 6;  
FILL 1 word fill prefix SBODEF tag $$;  
LPORT1 character length 4;  
NXT SYSID byte unsigned dimension 6;  
FILL 1 word fill prefix SBODEF tag $$;  
constant 'LENGTH' equals . prefix SBO$ tag K;  
constant 'LENGTH' equals . prefix SBO$ tag C;
```

```
end SBODEF;
```

```
end_module $SBODEF;
```

```
/*SYSTEM ID  
/*RESERVED WORD  
/*MAXIMUM DG SIZE  
/*MAXIMUM MSG SIZE  
/*SW TYPE, 1-4 CHAR, BLNK FILL  
/*SW VERSION, 1-4 CHAR, BLNK FILL  
/*SW INCARNATION #  
/*HW TYPE, 1-4 CHAR BLNK FILL  
/*HW VERSION, 1-4 CHAR BLNK FILL  
/*NODE NAME, COUNTED ASCII STRING  
/*START OF 12 BYTE SPECIFIER OF  
/* 1ST VC (PATH BLK) TO SYSTEM  
/*REMOTE STATION OF 1ST VC  
/*RESERVED WORD  
/*LOCAL PORT NAME OF 1ST VC  
/*ID OF NEXT SYSTEM IN CONFIGURATION  
/*RESERVED WORD  
/*LENGTH OF SBO ARRAY  
/*LENGTH OF SBO ARRAY
```

```
module $SCSDEF;
```

```
/*  
/* SCS MESSAGE DEFINITIONS  
/*  
/* THIS STRUCTURE DEFINES OFFSETS AND FIELDS WITHIN THE SCS PORTION OF  
/* A CLUSTER MESSAGE. OFFSETS ARE DEFINED RELATIVE TO THE START OF THE  
/* APPLICATION DATA OR SCS CONTROL MESSAGE DATA. THE FULL MESSAGE FORMAT  
/* CONSISTS OF A PORT DRIVER LAYER HEADER (SEE STRUCTURE PPD) FOLLOWED  
/* BY THE SCS HEADER LAYER FOLLOWED BY THE APPLICATION DATA OR SCS CONTROL  
/* MESSAGE DATA.  
/*--
```

```
aggregate SCSDEF structure prefix SCSS origin MIN_CR;  
  PPD byte unsigned dimension 16;  
  "LENGTH" word unsigned;
```

```
constant OVHD equals 14 prefix SCS tag $C;  
constant CON_REQ equals 66 prefix SCS tag $C;  
constant CON_RSP equals 18 prefix SCS tag $C;  
constant ACCP_REQ equals 66 prefix SCS tag $C;  
constant ACCP_RSP equals 18 prefix SCS tag $C;  
constant REJ_REQ equals 18 prefix SCS tag $C;  
constant REJ_RSP equals 14 prefix SCS tag $C;  
constant DISC_REQ equals 18 prefix SCS tag $C;  
constant DISC_RSP equals 14 prefix SCS tag $C;  
constant CR_REQ equals 18 prefix SCS tag $C;  
constant CR_RSP equals 14 prefix SCS tag $C;  
FILL 1 word fill prefix SCSDEF tag $$;  
MYPE word unsigned;
```

```
constant(  
  CON_REQ  
  , CON_RSP  
  , ACCP_REQ  
  , ACCP_RSP  
  , REJ_REQ  
  , REJ_RSP  
  , DISC_REQ  
  , DISC_RSP  
  , CR_REQ  
  , CR_RSP  
  , APPL_MSG  
  , APPL_DG  
  ) equals 0 increment 1 prefix SCS tag $C;  
CREDIT word unsigned;  
DST_CONID longword unsigned;  
SRC_CONID longword unsigned;  
constant APPL_BASE equals . prefix SCSS tag K;  
constant APPL_BASE equals . prefix SCSS tag C;  
MIN_CR word unsigned;
```

```
/*16 BYTES OF PPD HEADER  
/*MESSAGE LENGTH (INCLUDES ALL  
/* BYTES FROM SCSSW LENGTH ON,  
/* NOT INCLUDING SCSSW LENGTH)  
/* (FIELD SHARED BY PPD)  
/*DEFINE LENGTHS OF SCS CONTROL MSGS:  
/* SCS LAYER OVERHEAD  
/* CONNECT_REQ LENGTH  
/* CONNECT_RSP LENGTH  
/* ACCEPT_REQ LENGTH  
/* ACCEPT_RSP LENGTH  
/* REJECT_REQ LENGTH  
/* REJECT_RSP LENGTH  
/* DISCONNECT_REQ LENGTH  
/* DISCONNECT_RSP LENGTH  
/* CREDIT_REQ LENGTH  
/* CREDIT_RSP LENGTH  
/*WORD RESERVED FOR PPD LAYER  
/*SCS MESSAGE TYPE  
/*SCS MESSAGE TYPE CODES:  
/* 0 ORIGIN, INCREMENTS OF 1
```

```
/* CONNECT_REQ  
/* CONNECT_RSP  
/* ACCEPT_REQ  
/* ACCEPT_RSP  
/* REJECT_REQ  
/* REJECT_RSP  
/* DISCONNECT_REQ  
/* DISCONNECT_RSP  
/* CREDIT_REQ  
/* CREDIT_RSP  
/* APPLICATION MESSAGE  
/* APPLICATION DATAGRAM
```

```
/*CREDIT BEING EXTENDED  
/*DESTINATION (RECVING) CONNX ID  
/*SOURCE (SENDING) CONNX ID  
/*BASE OF APPLICATION MESSAGE DATA  
/*BASE OF APPLICATION MESSAGE DATA  
/*MINIMUM SEND CREDIT
```

```
/*  
/*  
/*
```

```
aggr
```

```
end;
```

```
/*  
/*  
/*
```

```
aggr
```

```
end;
```

```
/*  
/*  
/*
```

```
aggr
```

```
end;
```

STATUS word unsigned;

constant STNORMAL equals 1 prefix SCS\$ tag K;
 constant STNORMAL equals 1 prefix SCS\$ tag C;
 constant STNOMAT equals 10 prefix SCS\$ tag K;
 constant STNOMAT equals 10 prefix SCS\$ tag C;
 constant STNORS equals 18 prefix SCS\$ tag K;
 constant STNORS equals 18 prefix SCS\$ tag C;
 constant STDISC equals 25 prefix SCS\$ tag K;
 constant STDISC equals 25 prefix SCS\$ tag C;
 constant STINSFCR equals 33 prefix SCS\$ tag K;
 constant STINSFCR equals 33 prefix SCS\$ tag C;

constant CON_BASE equals . prefix SCS\$ tag K;
 constant CON_BASE equals . prefix SCS\$ tag C;

DST_PROC character length 16;
 SRC_PROC character length 16;
 CON_DAT byte unsigned dimension 16;

end SCSDEF;

/*
 /* DEFINITION OF THE REQUEST/SEND DATA OFFSETS
 /*

aggregate SCSDEF1 structure prefix SCS\$ origin SND_BOFF;

LCONID longword unsigned;
 RSPID longword unsigned;
 XCT_LEN longword unsigned;
 SND_NAME longword unsigned;
 SND_BOFF longword unsigned;
 REC_NAME longword unsigned;
 REC_BOFF longword unsigned;

end SCSDEF1;

end_module \$SCSDEF;

/*STATUS/REASON
 /*DEFINE STATUS/REASON CODES:
 /* NORMAL, SUCCESS
 /*
 /* NO MATCHING LISTENER
 /*
 /* NO RESOURCES
 /*
 /* DISCONNECTED
 /*
 /* INSUFF CREDIT
 /*
 /*BASE OF CONNECT/ACCEPT INFO TO
 /*BASE OF CONNECT/ACCEPT INFO TO
 /* GIVE TO SYSAP'S
 /*FORMAT OF CONNECT/ACCEPT REQ MSGS:
 /* DESTINATION PROCESS NAME
 /* SOURCE PROCESS NAME
 /* CONNECT DATA

```
module SCSCMGDEF;
```

```
/*
```

```
/* SCSCMG - SCS CONNECTION MANAGEMENT MESSAGE FORMAT
```

```
/*
```

```
/* THIS PORTION OF A CONNECT/ACCEPT MESSAGE IS SEEN BY A
```

```
/* SYSTEM APPLICATION.
```

```
/*-
```

```
aggregate SCSCMGDEF structure prefix SCSCMG$;
```

```
    RECNAM character length 16;
```

```
    SNDNAM character length 16;
```

```
    SNDDAT byte unsigned dimension 16;
```

```
/*RECEIVE PROCESS NAME
```

```
/*SENDER PROCESS NAME
```

```
/*SENDER CONNECT DATA
```

```
end SCSCMGDEF;
```

```
end_module SCSCMGDEF;
```

```
module $SDIRDEF;
```

```
/*
```

```
/* SDIR - SCS DIRECTORY ENTRY
```

```
/* THIS DATA STRUCTURE IS ALLOCATED FOR EACH LOCAL PROCESS THAT WANTS  
/* TO BE KNOWN TO SCS.
```

```
/*-
```

```
aggregate SDIRDEF structure prefix SDIR$;
```

```
FLINK longword unsigned;
```

```
BLINK longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYPE byte unsigned;
```

```
PROCNAM byte unsigned dimension 16;
```

```
PROCINF byte unsigned dimension 16;
```

```
CONID longword unsigned;
```

```
constant 'LENGTH' equals . prefix SDIR$ tag K;
```

```
constant 'LENGTH' equals . prefix SDIR$ tag C;
```

```
/*FWD LINK
```

```
/*BCK LINK
```

```
/*STRUCTURE SIZE IN BYTES
```

```
/*SCS STRUCTURE TYPE
```

```
/*SCS STRUCTURE SUBTYPE FOR SDIR
```

```
/*ASCII STRING FOR PROCESS NAME
```

```
/*ASCII STRING FOR PROCESS INFO
```

```
/*CONNECTION ID
```

```
end SDIRDEF;
```

```
end_module $SDIRDEF;
```

module \$SGNDEF;

/*+
/* SYSGEN PARAMETER DEFINITIONS
/*-

```
constant BALSETCNT equals 24 prefix SGN tag $C; /* NUMBER OF PROCESSES IN BALANCE SET
constant DFWSCNT equals 100 prefix SGN tag $C; /* DEFAULT WORKING SET COUNT
constant DFWSQUOTA equals 120 prefix SGN tag $C; /* DEFAULT WORKING SET QUOTA
constant GBLSECCNT equals 40 prefix SGN tag $C; /* GLOBAL SECTION COUNT
constant MAXGPGCNT equals 2*1024 prefix SGN tag $C; /* GLOBAL PAGE COUNT (GPT SIZE)
constant MAXPAGCNT equals 128*32*4 prefix SGN tag $C; /* PHYSICAL MEMORY SIZE IN PAGES
constant MAXPGFL equals 4096 prefix SGN tag $C; /* DEFAULT MAXIMUM PAGING FILE
constant MAXPSTCNT equals 5 prefix SGN tag $C; /* MAX NUMBER OF PST ENTRIES
constant MAXVPGCNT equals 8*8*128 prefix SGN tag $C; /* MAX PROCESS VIRTUAL SIZE (PAGES)
constant MAXWSCNT equals 1024 prefix SGN tag $C; /* MAX WORKING SET SIZE (PAGES)
constant MINWSCNT equals 10 prefix SGN tag $C; /* MIN WORKING SET SIZE (PAGES)
constant NPAGEDYN equals 52*512 prefix SGN tag $C; /* NON-PAGED DYNAMIC POOL SIZE
constant NPROCS equals 64 prefix SGN tag $C; /* MAX NUMBER OF PROCESSES
constant PAGEDYN equals 2*16*512 prefix SGN tag $C; /* PAGED DYNAMIC POOL SIZE IN BYTES
constant PHYPCGNT equals 32*128 prefix SGN tag $C; /* ACTUAL PHYSICAL PAGE COUNT
constant SYSDWSCNT equals 40 prefix SGN tag $C; /* DEFAULT SYSTEM WORKING SET COUNT
constant SYSVECPGS equals 5 prefix SGN tag $C; /* NO. OF PAGES OF SYSTEM SERVICE VECTORS
constant SYSWSCNT equals 96 prefix SGN tag $C; /* SYSTEM WORKING SET COUNT
```

end_module \$SGNDEF;

```
module $SHBDEF;
```

```
/*  
/* SHARED MEMORY CONTROL BLOCK DEFINITIONS  
/*  
/*  
/* The UETP for the MA780 depends on some of the following definitions. Please  
/* let someone in that group know if the definitions change substantially.  
/*
```

```
aggregate SHBDEF structure prefix SHB$;
```

```
  LINK longword unsigned;          /*LINK TO NEXT SHB  
  DATAPAGE longword unsigned;      /*VIRTUAL ADDRESS OF DATAPAGE  
  SIZE word unsigned;              /*SIZE OF SHB IN BYTES  
  TYPE byte unsigned;              /*STRUCTURE TYPE FOR SHB  
  FLAGS_OVERLAY union fill;  
    FLAGS byte unsigned;           /*FLAGS  
    FLAGS_BITS structure fill;  
      CONNECT bitfield mask;      /* MEMORY IS CONNECTED, USEABLE  
    end FLAGS_BITS;  
  end FLAGS_OVERLAY;  
  REFCNT longword unsigned;        /*COUNT OF REFERENCES TO MEMORY  
  BASGSPFN longword unsigned;      /*BASE PFN FOR GLOBAL SECTION PAGES  
  NEXUS byte unsigned;             /*NEXUS OF PORT  
  PORT byte unsigned;              /*PORT NUMBER  
  FILL 1 word fill prefix SHBDEF tag $$;  
  POOLEND longword unsigned;       /* UNUSED  
  ADP longword unsigned;           /*ADDRESS PAST LAST BYTE OF POOL  
  constant 'LENGTH' equals . prefix SHB$ tag K; /*ADAPTER CONTROL BLOCK ADDRESS  
  constant 'LENGTH' equals . prefix SHB$ tag C; /*LENGTH OF CONTROL BLOCK  
                                     /*LENGTH OF CONTROL BLOCK
```

```
end SHBDEF;
```

```
end_module $SHBDEF;
```

```
module $SHDDEF;
```

```
/*  
/* SHARED MEMORY DATAPAGE DEFINITIONS  
/*-  
/*
```

```
/* The UETP for the MA780 depends on some of the following definitions. Please  
/* Let someone in that group know if the definitions change substantially.  
/*
```

```
constant MAXPORTS equals 16 prefix SHD tag $C; /*MAXIMUM NUMBER PORTS HANDLED BY THIS  
/*DATA STRUCTURE
```

```
/**** START OF CONSTANT FIELDS:
```

```
aggregate SHDDEF structure prefix SHD$;
```

```
MBXPTR longword unsigned; /*RELATIVE POINTER TO MAILBOX TABLE  
GSDPTR longword unsigned; /*RELATIVE POINTER TO GSD TABLE  
CEFPTR longword unsigned; /*RELATIVE POINTER TO CEF TABLE  
GSBITMAP longword unsigned; /*RELATIVE POINTER TO BITMAP  
GSPAGCNT longword unsigned; /*CNT OF PAGES ALLOTTED FOR GBL SECTIONS  
GSPFN longword unsigned; /*RELATIVE PFN OF 1ST GBL SECTION PAGE  
GSDMAX word unsigned; /*MAX GSD'S (SIZE OF TABLE)  
MBXMAX word unsigned; /*MAX MAILBOXES (SIZE OF TABLE)  
CEFMAY word unsigned; /*MAX CEF CLUSTERS (SIZE OF TABLE)  
FILL_1 word fill prefix SHDDEF tag $$; /*UNUSED  
NAME character length 16; /*NAME OF MEMORY (COUNTED STRING)  
constant NAMELENGTH equals 16 prefix SHD tag $C; /*MAXIMUM LENGTH OF NAME OF MEMORY  
INITTIME quadword unsigned; /*INITIALIZATION TIME
```

```
/**** END OF CONSTANT FIELDS.
```

```
CRC longword unsigned; /*CRC OF CONSTANT FIELDS  
GSDQUOTA word unsigned dimension 16; /*COUNT OF GSD'S CREATED (ONE/PORT)  
MBXQUOTA word unsigned dimension 16; /*COUNT OF MAILBOXES CREATED (ONE/PORT)  
CEFQUOTA word unsigned dimension 16; /*COUNT OF CLUSTERS CREATED (ONE/PORT)  
PORTS byte unsigned; /*NUMBER OF PORTS  
INITLCK byte unsigned; /*OWNER OF INIT LOCK  
BITMAPLCK byte unsigned; /*OWNER OF GS BITMAP LOCK  
FLAGS_OVERLAY union fill;  
  FLAGS byte unsigned; /*FLAGS FOR LOCKING DATA STRUCTURES  
  FLAGS_BITS structure fill;  
    INITLCK bitfield mask; /*COMMON DATA PAGE BEING INITIALIZED  
    BITMAPLCK bitfield mask; /*BITMAP BEING MODIFIED  
    GSDLCK bitfield mask; /*GLOBAL SECTION DSC TABLE BEING SEARCHED  
    MBXLCK bitfield mask; /*MAILBOX TABLE BEING SEARCHED  
    CEFCLK bitfield mask; /*COMMON EVENT FLAG TABLE BEING SEARCHED
```

```
end FLAGS_BITS;
```

```
end FLAGS_OVERLAY;
```

```
GSDLOCK byte unsigned;  
MBXLOCK byte unsigned;  
CEFLOCK byte unsigned;  
FILL_2 byte fill prefix SHDDEF tag $$;  
PROWAIT word unsigned;
```

```
POLL word unsigned;
```

```
RESWAIT word unsigned dimension 16;
```

```
/*OWNER OF GSD TABLE LOCK  
/*OWNER OF MBX TABLE LOCK  
/*OWNER OF CEF TABLE LOCK  
/*UNUSED  
/*PORTS WAITING FOR INTER-PROCESSOR REQUEST BLOCKS  
/* (ONE BIT/PORT)  
/*PORTS ACTIVELY USING THE MEMORY  
/* (ONE BIT/PORT)  
/*PORTS WAITING FOR A RESOURCE  
/* (ONE BIT/PORT, ONE MASK/RESOURCE)
```

```
modi
```

```
/*  
/* 1  
/*  
/*
```

```
aggi
```

```
end
```

```
end.
```

```
RESAVAIL word unsigned dimension 16;          /*PORTS NEEDING TO REPORT RESOURCE AVAILABLE
RESSUM word unsigned;                        /* (ONE BIT/PORT, ONE MASK/RESOURCE)
FILL_3 word fill prefix SHDDEF tag $$;        /*PORTS WITH RESOURCES TO REPORT
FILL_4 longword fill prefix SHDDEF tag $$;    /* (ONE BIT/PORT)
/**** NOTE: THE FOLLOWING FIELDS MUST BE QUADWORD ALIGNED:
PRQ quadword unsigned;                       /*UNUSED
POOL quadword unsigned;                      /*UNUSED
PRQWRK quadword unsigned dimension 16;        /*FREE INTER-PROCESSOR REQUEST BLOCK LISTHEAD
                                              /*FREE POOL BLOCK LISTHEAD
                                              /*INTER-PROCESSOR REQUEST WORK QUEUE LISTHEADS
                                              /* (ONE LISTHEAD/PORT)
constant 'LENGTH' equals . prefix SHD$ tag K; /*LENGTH OF DATAPAGE
constant 'LENGTH' equals . prefix SHD$ tag C; /*LENGTH OF DATAPAGE
end SHDDEF;
end_module $SHDDEF;
```

```
module $SHLDEF;
```

```
/*
```

```
/* SHL - SHAREABLE IMAGE LIST
```

```
/*
```

```
/* THIS LIST IS CREATED IN THE IMAGE FIXUP SECTION BY THE LINKER AND
```

```
/* USED BY THE IMAGE ACTIVATOR FOR DOING SHAREABLE IMAGE FIXUPS.
```

```
/*-
```

```
aggregate SHLDEF structure prefix SHLS;
```

```
BASEVA longword unsigned;
```

```
SHLPTR longword unsigned;
```

```
"IDENT" longword unsigned;
```

```
PERMCTX longword unsigned;
```

```
SHL_SIZE byte unsigned;
```

```
FILL_1 byte fill prefix SHLDEF tag $$;
```

```
FILL_2 word fill prefix SHLDEF tag $$;
```

```
FILL_3 longword fill prefix SHLDEF tag $$;
```

```
constant OLD_SHL_SIZE equals 56 prefix SHL tag $C;
```

```
constant MAXNAMLEN equals 39 prefix SHL tag $C;
```

```
IMGNAM_OVERLAY union fill;
```

```
    IMGNAM character length 40;
```

```
    constant 'LENGTH' equals . prefix SHLS tag K;
```

```
    constant 'LENGTH' equals . prefix SHLS tag C;
```

```
    NAMLEN byte unsigned;
```

```
end IMGNAM_OVERLAY;
```

```
end SHLDEF;
```

```
end_module $SHLDEF;
```

```
/* Base address of this shareable image
```

```
/* Pointer from SHL in shareable image
```

```
/* to associated SHL in executable image
```

```
/* GSMATCH
```

```
/* Permanent sharable image context
```

```
/* Size of SHL elements
```

```
/* Spare for extensions
```

```
/* Spare for extensions
```

```
/* Spare for extensions
```

```
/* Size of 'old' SHL
```

```
/* Maximum length of image name
```

```
/* Shareable image name (ASCII string)
```

```
/* Length of shareable image list element
```

```
/* Length of shareable image list element
```

```
/* Synonym for name count
```

```
module $SLVDEF;
```

```
/*
/*
/* Define symbolic offsets for System Loadable Vectors. These symbols
/* are used by the various pieces of the loadable EXEC, notably SCSVEC,
/* to create a list of vectors in system space and a corresponding image
/* that will be loaded into pool and connected to the system vectors.
/*
/*
```

```
aggregate SLV structure prefix SLV$;
```

```
  CODESIZE longword unsigned; /* Loadable image size (in bytes)
  INITRTN longword unsigned; /* Offset to init. routine
  SIZE word unsigned; /* Same as SLV$CODESIZE
  TYPE byte unsigned; /* Structure type (DYN$C_LOADCODE)
  SUBTYP byte unsigned; /* Structure Subtype
  PROT_R byte unsigned; /* writeable protection for image
  PROT_W byte unsigned; /* read-only protection for image
  SPARE word unsigned; /* spare field for future use
  SYSVECS address; /* address of vectors in SYS.EXE
  FACILITY character length 16; /* facility name (.ASCII)
  LIST character length 640; /* Start of vector list (MAXVEC*5)
constant 'LENGTH' equals .; /* SLV$K_LENGTH
end SLV;
```

```
/*
/*
/* Define vector type codes. The codes LODUMMY and HIDUMMY are
/* used as placeholders, to make the definition of the upper and
/* lower bound vector type symbols automatic. New vector type codes
/* should be added at the end of the list, but before HIDUMMY.
/*
/*
```

```
constant (LODUMMY, /*
  LDATA, /* Longword pointer to data
  AJUMP, /* Aligned jump
  UJUMP, /* Unaligned jump
  SDATA, /* Specified data
  SJUMP, /* Specified jump
  HIDUMMY
)
```

```
  equals 0 increment 1 prefix SLV tag $K;
constant MINTYPE equals SLV$K_LODUMMY+1 prefix SLV tag $K; /* Lower bound of vector type codes
constant MAXTYPE equals SLV$K_HIDUMMY-1 prefix SLV tag $K; /* Upper bound of vector type codes
constant MAXVEC equals 128 prefix SLV tag $K; /* Max. # of vectors in list.
```

```
end_module $SLVDEF;
```

```
module $SMBDEF;                /* Symbiont interface definitions
```

```
/*+
/* Symbolic definitions for the symbiont to job controller interface.
```

```
/*
/* Public definitions of message types, item codes, and
/* other constants utilized by the symbiont to job controller
/* interface facility.
```

```
/*-
```

```
/*
/* Structure level
/*
```

```
constant SMBMSG$K_STRUCTURE_LEVEL equals 1; /* Current structure level
constant SMBMSG$K_STRUCTURE_LEVEL_1 equals 1; /* Structure level 1
```

```
/*
/* Request header
/*
```

```
aggregate REQUEST_HEADER structure prefix SMBMSG$;
```

```
    REQUEST_CODE      word unsigned; /* Request code
```

```
    constant (
        PAUSE_TASK, /* Define request codes
        RESET_STREAM, /* - STOP /QUEUE
        RESUME_TASK, /* - STOP /QUEUE /RESET
        START_STREAM, /* - START /QUEUE (when paused)
        START_TASK, /* - START /QUEUE (when stopped)
        STOP_STREAM, /* - task available
        STOP_TASK, /* - STOP /QUEUE /NEXT
        TASK_COMPLETE, /* - STOP /QUEUE /ABORT or /REQUEUE
        TASK_STATUS, /* - stream is idle
        MAX_REQUEST_CODE /* - asynchronous status update
    ) equals 1 increment 1; /* MUST BE LAST
```

```
    STRUCTURE_LEVEL   byte unsigned; /* Message structure level
    STREAM_INDEX       byte unsigned; /* Stream index
```

```
end;
```

```
/*
/* Item header
/*
```

```
aggregate ITEM_HEADER structure prefix SMBMSG$;
```

```
    ITEM_SIZE          word unsigned; /* Item size
    ITEM_CODE           word unsigned; /* Item code
```

```
    constant (
        ACCOUNTING_DATA, /* Define item codes
        ACCOUNT_NAME, /* - accounting information
        /* - account name
```

AFTER TIME,	/* - /AFTER value
ALIGNMENT PAGES,	/* - /ALIGN count
BOTTOM MARGIN,	/* - trailing blank lines
CHARACTERISTICS,	/* - /CHARACTERISTICS value
CHECKPOINT DATA,	/* - checkpoint information
CONDITION VECTOR,	/* - task error messages
DEVICE_NAME,	/* - /DN value
DEVICE STATUS,	/* - device status
ENTRY NUMBER,	/* - job entry number
EXECUTOR QUEUE,	/* - this output queue
FILE_COPIES,	/* - /COPIES value
FILE_COUNT,	/* - current file copy number
FILE_SETUP_MODULES,	/* - file setup module list
FIRST_PAGE,	/* - first page to print
FORM_LENGTH,	/* - lines per page
FORM_NAME,	/* - name of physical form
FORM_SETUP_MODULES,	/* - form setup module list
FORM_WIDTH,	/* - columns per line
FILE_IDENTIFICATION,	/* - device, fid, and did
FILE_SPECIFICATION,	/* - file name
JOB_COPIES,	/* - /JOB_COUNT value
JOB_COUNT,	/* - current job copy number
JOB_NAME,	/* - /NAME value
JOB_RESET_MODULES,	/* - job reset module list
LAST_PAGE,	/* - last page to print
LEFT_MARGIN,	/* - leading blank columns
LIBRARY_SPECIFICATION,	/* - library name
MAXIMUM_STREAMS,	/* - maximum supported symbiont
MESSAGE_VECTOR,	/* - error messages to print
NOTE,	/* - /NOTE value
PAGE_SETUP_MODULES,	/* - page setup module list
PARAMETER_1,	/* - user parameter 1
PARAMETER_2,	/* - user parameter 2
PARAMETER_3,	/* - user parameter 3
PARAMETER_4,	/* - user parameter 4
PARAMETER_5,	/* - user parameter 5
PARAMETER_6,	/* - user parameter 6
PARAMETER_7,	/* - user parameter 7
PARAMETER_8,	/* - user parameter 8
PRINT CONTROL,	/* - printing control
PRIORITY,	/* - queue priority
QUEUE,	/* - generic queue name
REFUSE REASON,	/* - reason task refused
RELATIVE_PAGE,	/* - /BACKWARD, /FORWARD values
REQUEST CONTROL,	/* - request control
REQUEST_RESPONSE,	/* - request code being responded to
RIGHT_MARGIN,	/* - trailing blank columns
SEARCH STRING,	/* - /SEARCH value
SEPARATION CONTROL,	/* - separation control
STOP_CONDITION,	/* - reason for print abort
TIME_QUEUED,	/* - time queued
TOP_MARGIN,	/* - leading blank lines
UIC,	/* - UIC of submitter
USER_NAME,	/* - username
	/*
MAX_ITEM_CODE	/* MUST BE LAST

) equals 1 increment 1;

end;

```
/*
/* ACCOUNTING_DATA item
/*
```

aggregate ACCOUNTING_DATA structure prefix SMBMSG\$;

PAGES_PRINTED	longword unsigned;	/* Pages printed
qio_puts	longword unsigned;	/* Lines printed
rms_gets	longword unsigned;	/* File reads
CPU_TIME	longword unsigned;	/* Processor time

#s_accounting_data = .;

end;

```
/*
/* CHECKPOINT_DATA item
/*
```

aggregate CHECKPOINT_DATA structure prefix SMBMSG\$;

FILLER	byte unsigned;	/* Reserved
CHECKPOINT_LEVEL	byte unsigned;	/* Checkpoint structure level
OFFSET	word unsigned;	/* Offset into record
CARCOM	longword unsigned;	/* Carriage control
PAGE	longword unsigned;	/* Page number
RECORD_NUMBER	longword unsigned;	/* Record number
USER_KEY	quadword;	/* User positioning key

#s_checkpoint_data = .;

end;

```
/*
/* DEVICE_STATUS item
/*
```

aggregate DEVICE_STATUS structure prefix SMBMSG\$;

DEVICE_FLAGS	structure longword unsigned;	/* Device flags
LOWERCASE	bitfield mask;	/* - supports lowercase
PAUSE_TASK	bitfield mask;	/* - symbiont initiated pause
REMOTE	bitfield mask;	/* - device is remote
SERVER	bitfield mask;	/* - server symbiont
STALLED	bitfield mask;	/* - task stalled
STOP_STREAM	bitfield mask;	/* - symbiont requesting stop stream
TERMINAL	bitfield mask;	/* - device is a terminal
UNAVAILABLE	bitfield mask;	/* - device unavailable
FILLER	bitfield length 32-^ fill;	/* - force longword

end;

```
/*
/* PRINT_CONTROL item
/*
```

```
aggregate PRINT_CONTROL structure prefix SMBMSG$;
PRINT_FLAGS- structure longword unsigned; /* Print flags
DOUBLE SPACE bitfield mask; /* - double space
PAGE_HEADER bitfield mask; /* - print page headers
PAGINATE bitfield mask; /* - insert <ff>'s
PASSALL bitfield mask; /* - binary print file
SEQUENCED bitfield mask; /* - print sequence numbers
SHEET_FEED bitfield mask; /* - pause at every TOF
TRUNCATE bitfield mask; /* - truncate on overflow
WRAP bitfield mask; /* - wrap on overflow
FILLER bitfield length 32-^ fill; /* - force longword
end;
end;
```

```
/*
/* REQUEST_CONTROL item
/*
```

```
aggregate REQUEST structure prefix SMBMSG$;
REQUEST_FLAGS- structure longword unsigned; /* Print flags
ALIGNMENT_MASK bitfield mask; /* - print A's and 9's
PAUSE_COMPLETE bitfield mask; /* - pause when request complete
RESTARTING bitfield mask; /* - job is restarting
TOP_OF_FILE bitfield mask; /* - rewind before resume
FILLER- bitfield length 32-^ fill; /* - force longword
end;
end;
```

```
/*
/* SEPARATION_CONTROL item
/*
```

```
aggregate SEPARATION_CONTROL structure prefix SMBMSG$;
SEPARATION_FLAGS- structure longword unsigned; /* Print flags
FILE_BURST bitfield mask; /* - print file burst page
FILE_FLAG bitfield mask; /* - print file flag page
FILE_TRAILER bitfield mask; /* - print file trailer page
FILE_TRAILER_ABORT bitfield mask; /* - print file trailer page
JOB_FLAG bitfield mask; /* - print job flag page
JOB_BURST bitfield mask; /* - print job burst page
JOB_RESET bitfield mask; /* - execute job reset sequence
JOB_RESET_ABORT bitfield mask; /* - execute job reset sequence
JOB_TRAILER bitfield mask; /* - print job trailer page
JOB_TRAILER_ABORT bitfield mask; /* - print job trailer page
FILLER bitfield length 32-^ fill; /* - force longword
end;
end;
```

SYSDEFQZ.SDL;1

16-SEP-1984 16:45:41.35 Page 46

end_module \$SMBDEF;

SYS

```
module $SPNBDEF;
```

```
/*
```

```
/* SPNB - SCA POLLER NAME BLOCK
```

```
/* THIS DATA STRUCTURE CONTAINS A LIST OF PROCESS NAMES WHICH WILL  
/* BE SEARCHED FOR ON THE GIVEN REMOTE NODE.
```

```
/*-
```

```
aggregate SPNBDEF structure prefix SPNB$;
```

```
FLINK longword unsigned;
```

```
BLINK longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYP byte unsigned;
```

```
SB longword unsigned;
```

```
ROUTINE longword unsigned;
```

```
INDEX byte unsigned;
```

```
REFC word unsigned;
```

```
FREE byte unsigned dimension 1;
```

```
constant 'HDRSZ' equals . prefix SPNB$ tag C;
```

```
NAMLST byte unsigned;
```

```
/*FWD LINK
```

```
/*BCK LINK
```

```
/*STRUCTURE SIZE IN BYTES
```

```
/*SCS STRUCTURE TYPE
```

```
/*SCS STRUCTURE SUBTYPE FOR SPNB
```

```
/*SYSTEM BLOCK OF REMOTE NOTE
```

```
/*ADDRESS OF ROUTINE TO BE CALLED WHEN PROCESS FOUND
```

```
/*INDEX INTO PROCESS LIST OF NEXT PROCESS TO SEARCH FOR
```

```
/*NUMBER OF REFERENCES TO SPNB
```

```
/*FREE BYTE
```

```
/*SIZE OF HEADER
```

```
/*START OF VARIABLE LENGTH LIST OF ADDRESSES OF PROCESS NAMES
```

```
/*LIST IS ZERO TERMINATED
```

```
end SPNBDEF;
```

```
end_module $SPNBDEF;
```

```
module $SPPBDEF;
```

```
/*
```

```
/* SPPB - SCA POLLER PROCESS BLOCK
```

```
/*
```

```
/* THIS DATA STRUCTURE DESCRIBES A PROCESS NAME KNOWN
```

```
/* TO THE SCA DIRECTORY POLLER.
```

```
/*-
```

```
aggregate SPPBDEF structure prefix SPPB$;
```

```
FLINK longword unsigned;
```

```
BLINK longword unsigned;
```

```
SIZE word unsigned;
```

```
TYPE byte unsigned;
```

```
SUBTYP byte unsigned;
```

```
PROCNAM byte unsigned dimension 16;
```

```
RTN longword unsigned;
```

```
CTX longword unsigned;
```

```
BIT word unsigned;
```

```
UNUSED_1 word unsigned fill;
```

```
constant 'LENGTH' equals : prefix SPPB$ tag K;
```

```
constant 'LENGTH' equals : prefix SPPB$ tag C;
```

```
/*FWD LINK
```

```
/*BCK LINK
```

```
/*STRUCTURE SIZE IN BYTES
```

```
/*SCS STRUCTURE TYPE
```

```
/*SCS STRUCTURE SUBTYPE FOR SPPB
```

```
/*ASCII STRING FOR PROCESS NAME
```

```
/*ADDRESS OF NOTIFICATION ROUTINE
```

```
/*CONTEXT FOR NOTIFICATION ROUTINE
```

```
/*BIT ASSIGNED TO THIS PROCESS NAME
```

```
/*WORD RESERVED FOR EXPANSION
```

```
end SPPBDEF;
```

```
end_module $SPPBDEF;
```

mod

/*

/*

/*-

agg

end

agg

end

agg

end

agg

end

end

module \$STATEDEF;

/*+
/* SCHEDULING STATES
/*-

```
constant(  
  COLPG  
  , MWAIT  
  , CEF  
  , PFW  
  , LEF  
  , LEFO  
  , HIB  
  , HIBO  
  , SUSP  
  , SUSPO  
  , FPG  
  , COM  
  , COMO  
  , CUR  
) equals 1 increment 1 prefix SCH tag $C;
```

end_module \$STATEDEF;

/* DEFINITIONS START AT 1

```
/*COLLIDED PAGE WAIT  
/*MUTEX AND MISCELLANEOUS RESOURCE WAIT  
/*COMMON EVENT FLAG WAIT STATE  
/*PAGE FAULT WAIT  
/*LOCAL EVENT FLAG WAIT  
/*LOCAL EVENT FLAG WAIT OUT OF BALANCE SET  
/*HIBERNATE WAIT  
/*HIBERNATE WAIT OUT OF BALANCE SET  
/*SUSPENDED  
/*SUSPENDED OUT OF THE BALANCE SET  
/*FREEPAGE WAIT  
/*COMPUTE, IN BALANCE SET STATE  
/*COMPUTE, OUT OF BALANCE SET STATE  
/*CURRENT PROCESS STATE
```

```
module $SYSAPDEF;
```

```
/*+
/* SYSAP - FLAGS USED IN THE SYSAP-SCS INTERFACE
/*-
```

```
constant (
    DISPO
    , DISPRET
    DISPPO
) equals 0 increment 1 prefix SYSAP tag $C;
```

```
constant (
    DGREC
    DGSNT
) equals 0 increment 1 prefix SYSAP tag $C;
```

```
end_module $SYSAPDEF;
```

```
/*OPTIONS FOR DISPOSING OF
/* SENT DATAGRAM:
/* 0 ORIGIN, INCR OF 1:
```

```
/* DISPOSE ON DG FREE QUEUE
/* DISPOSE BY RETURN TO SYSAP
/* DISPOSE BY RETURN TO POOL
```

```
/*FLAGS SPECIFYING TYPE OF DG
/* REC'D FROM REMOTE SYSAP:
/* 0 ORIGIN, INCR OF 1:
```

```
/* DG REC'D FROM REMOTE
/* DG SENT
```

```
module $TASTDEF;
/*
/* TERMINAL AST PACKET. THIS STRUCTURE IS USED BY TERMINAL SERVICES TO
/* DELIVER OUT OF BAND CHARACTER ASTS.
/*

aggregate TASTDEF structure prefix TAST$:
  FILL 1 longword dimension 7 fill prefix TASTDEF tag $$; /*RESERVE ACB REGION
  FLINK longword unsigned; /*FORWARD LINK
  AST longword unsigned; /*SAVED AST ADDRESS
  ASTPRM longword unsigned; /*SAVED AST PARAMETER
  PID longword unsigned; /*SAVED PID
  RMOD byte unsigned; /*SAVED RMOD
  CTRL OVERLAY union fill;
    CTRL byte unsigned; /*CONTROL FIELD
    CTRL BITS structure fill;
      MASK DSBL bitfield mask; /*DISABLE MASK PROCESSING
      INCLUDE bitfield mask; /*INCLUDE CHARACTER
      ONE SHOT bitfield mask; /*ONE SHOT AST
      BUSY bitfield mask; /*BLOCK BUSY
      LOST bitfield mask; /*AST LOST
      ABORT bitfield mask; /*ABORT I/O
    end CTRL BITS;
  end CTRL OVERLAY;
  CHAN word unsigned; /*CHANNEL
  MASK longword unsigned; /*OUT OF BAND MASK
  constant 'LENGTH' equals . prefix TAST$ tag K;
  constant 'LENGTH' equals . prefix TAST$ tag C;

  STATUS BITS structure;
    FILL bitfield length 14; /* First byte and spares
    ABO bitfield mask; /* ABORT flag
    INC bitfield mask; /* INCLUDE flag
  end STATUS_BITS;

end TASTDEF;
end_module $TASTDEF;
```

```
module STQDEF;
```

```
/*
```

```
/* TQE - TIME QUEUE ENTRY
```

```
/*
```

```
/* TIME QUEUE ENTRIES ARE UTILIZED TO SET TIMERS, WAKE UP PROCESSES, AND
```

```
/* FOR INTERNAL SYSTEM SUBROUTINES.
```

```
/*-
```

```
aggregate TQDEF structure prefix TQES;
```

```
  TQFL longword unsigned;
```

```
  TQBL longword unsigned;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  RQTYPE OVERLAY union fill;
```

```
    RQTYPE byte unsigned;
```

```
    RQTYPE BITS structure fill;
```

```
      FILL 1 bitfield length 2 fill prefix TQDEF tag $$; /* STARTING OFFSET
```

```
      REPEAT bitfield mask; /* REPEAT REQUEST (1=YES)
```

```
      ABSOLUTE bitfield mask; /* Absolute expiration time specified
```

```
    end RQTYPE BITS;
```

```
  end RQTYPE OVERLAY;
```

```
  PID_OVERLAY union fill;
```

```
    PID longword unsigned;
```

```
    FPC longword unsigned;
```

```
  end PID_OVERLAY;
```

```
  AST_OVERLAY union fill;
```

```
    AST longword unsigned;
```

```
    FR3 longword unsigned;
```

```
  end AST_OVERLAY;
```

```
  ASTPRM_OVERLAY union fill;
```

```
    ASTPRM longword unsigned;
```

```
    FR4 longword unsigned;
```

```
  end ASTPRM_OVERLAY;
```

```
  TIME quadword unsigned;
```

```
  DELTA quadword unsigned;
```

```
  RMOD byte unsigned;
```

```
  EFN byte unsigned;
```

```
  FILL 2 word fill prefix TQDEF tag $$;
```

```
  RQPID longword unsigned;
```

```
  constant 'LENGTH' equals . prefix TQES tag K;
```

```
  constant 'LENGTH' equals . prefix TQES tag C;
```

```
/*TIME QUEUE FORWARD LINK
```

```
/*TIME QUEUE BACKWARD LINK
```

```
/*SIZE OF TQE IN BYTES
```

```
/*STRUCTURE TYPE FOR TQE
```

```
/*TIME QUEUE ENTRY TYPE
```

```
/*TIMER OR WAKE REQUEST PROCESS ID
```

```
/*TIMER SUBROUTINE ADDRESS
```

```
/*ADDRESS OF AST ROUTINE
```

```
/*TIMER SUBROUTINE SAVED R3
```

```
/*AST PARAMETER
```

```
/*TIMER SUBROUTINE SAVED R4
```

```
/*ABSOLUTE EXPIRATION TIME
```

```
/*DELTA REPEAT TIME
```

```
/*ACCESS MODE OF REQUEST
```

```
/*EVENT FLAG NUMBER AND EVENT GROUP
```

```
/*SPARE WORD
```

```
/*REQUESTER PROCESS ID
```

```
/*LENGTH OF STANDARD TQE
```

```
/*LENGTH OF STANDARD TQE
```

```
/*
```

```
/* TIME QUEUE ENTRY REQUEST TYPE DEFINITIONS
```

```
/*
```

```
  constant TMSNGL equals 0 prefix TQE tag $C; /*TIMER ENTRY SINGLE SHOT REQUEST
```

```
  constant SSREPT equals 1+TQESM REPEAT prefix TQE tag $C; /*SYSTEM SUBROUTINE REPEAT REQUEST
```

```
  constant SSSNGL equals 1 prefix TQE tag $C; /*SYSTEM SUBROUTINE SINGLE SHOT REQUEST
```

```
  constant WKREPT equals 2+TQESM REPEAT prefix TQE tag $C; /*WAKE ENTRY REPEAT REQUEST
```

```
  constant WKSNGL equals 2 prefix TQE tag $C; /*WAKE ENTRY SINGLE SHOT REQUEST
```

```
end TQDEF;
```

SYSDEFQZ.SDL;1

16-SEP-1984 16:45:41.33¹³ Page 53

end_module \$TQDEF;

SYS

```
module SUADEF;
```

```
/*++
```

```
/* User authorization file format
```

```
/* Note: With the exception of the username and account name,
```

```
/* all strings are blank padded counted strings. Username and
```

```
/* account name are uncounted, blank padded.
```

```
/*--
```

```
aggregate UAFDEF structure prefix UAF$;
```

```
  RTYPE byte unsigned;
```

```
  constant (
```

```
    USER_ID
```

```
  ) equals 1 increment 1 tag C;
```

```
  VERSION byte unsigned;
```

```
  constant (
```

```
    VERSION1
```

```
  ) equals 1 increment 1 tag C;
```

```
  USRDATAOFF word unsigned;
```

```
  USERNAME structure character length 32;
```

```
    FILL 0 character length 31 fill;
```

```
    USERNAME_TAG character;
```

```
  end USERNAME;
```

```
  UIC structure longword unsigned;
```

```
    MEM word unsigned;
```

```
    GRP word unsigned;
```

```
  end UIC;
```

```
  SUB_ID longword unsigned;
```

```
  PARENT_ID quadword unsigned;
```

```
  constant KEYED_PART equals . tag C;
```

```
  ACCOUNT character length 32;
```

```
  OWNER character length 32;
```

```
  DEFDEV character length 32;
```

```
  DEFDIR character length 64;
```

```
  LGICMD character length 64;
```

```
  DEFCLI character length 32;
```

```
  CLITABLES character length 32;
```

```
  PWD structure quadword unsigned;
```

```
    PWD longword unsigned;
```

```
  end PWD;
```

```
  PWD2 quadword unsigned;
```

```
  LOGFAILS word unsigned;
```

```
  SALT word unsigned;
```

```
  ENCRYPT byte unsigned;
```

```
  constant (
```

```
    AD II
```

```
  , PURDY
```

```
  , PURDY_V
```

```
  ) equals 0 increment 1 tag C;
```

```
  ENCRYPT2 byte unsigned;
```

```
  PWD_LENGTH byte unsigned;
```

```
  FILE_1 byte dimension 1 fill tag $$;
```

```
  EXPIRATION quadword unsigned;
```

```
  PWD_LIFETIME quadword unsigned;
```

```
/* UAF record type
```

```
/* main user ID record
```

```
/* UAF format version
```

```
/* this version
```

```
/* offset of counted string of user data
```

```
/* username
```

```
/* tag to differentiate records
```

```
/* user ID code
```

```
/* member subfield
```

```
/* group subfield
```

```
/* user sub-identifier
```

```
/* identifier of owner of this account
```

```
/* ISAM keys come this far
```

```
/* account name
```

```
/* owner's name
```

```
/* default device
```

```
/* default directory
```

```
/* login command file
```

```
/* default command interpreter
```

```
/* user CLI tables
```

```
/* hashed password
```

```
/* 32 bit subfield
```

```
/* second password
```

```
/* count of login failures
```

```
/* random password salt
```

```
/* encryption algorithm
```

```
/* encryption codes
```

```
/* AUTODIN-II 32 bit crc code
```

```
/* Purdy polynomial over salted input
```

```
/* Purdy polynomial + variable length username
```

```
/* encryption algorithm for 2nd pwd
```

```
/* minimum password length
```

```
/* expiration date for account
```

```
/* password lifetime
```

```

PWD DATE quadword unsigned; /* date of password change
PWD2 DATE quadword unsigned; /* date of 2nd password change
LASTLOGIN_I quadword unsigned; /* date of last interactive login
LASTLOGIN_N quadword unsigned; /* date of last non-interactive login
PRIV quadword unsigned; /* process privilege vector
DEF_PRIV quadword unsigned; /* default process privileges
MIN_CLASS structure; /* minimum security class
    FILL 2 byte dimension 20 fill;
end MIN_CLASS;
MAX_CLASS structure; /* maximum security class
    FILL 3 byte dimension 20 fill;
end MAX_CLASS;
FLAGS structure longword unsigned; /* user flags longword
    DISCTLY bitfield; /* no user control-y
    DEFCLI bitfield; /* only allow user default CLI
    LOCKPWD bitfield; /* disable SET PASSWORD command
    CAPTIVE bitfield; /* captive account (no overrides)
    DISACNT bitfield; /* no interactive login
    DISWELCOM bitfield; /* skip welcome message
    DISMAIL bitfield; /* skip new mail message
    NOMAIL bitfield; /* disable mail delivery
    GENPWD bitfield; /* passwords must be generated
    PWD_EXPIRED bitfield; /* password has expired
    PWD2_EXPIRED bitfield; /* 2nd password has expired
    AUDIT bitfield; /* audit all actions
    DISREPORT bitfield; /* skip last login messages
    DISRECONNECT bitfield; /* inhibit reconnections
end FLAGS;
NETWORK_ACCESS_P byte unsigned dimension 3; /* hourly network access, primary
NETWORK_ACCESS_S byte unsigned dimension 3; /* hourly network access, secondary
BATCH_ACCESS_P byte unsigned dimension 3; /* hourly batch access, primary
BATCH_ACCESS_S byte unsigned dimension 3; /* hourly batch access, secondary
LOCAL_ACCESS_P byte unsigned dimension 3; /* hourly local access, primary
LOCAL_ACCESS_S byte unsigned dimension 3; /* hourly local access, secondary
DIALUP_ACCESS_P byte unsigned dimension 3; /* hourly dialup access, primary
DIALUP_ACCESS_S byte unsigned dimension 3; /* hourly dialup access, secondary
REMOTE_ACCESS_P byte unsigned dimension 3; /* hourly remote access, primary
REMOTE_ACCESS_S byte unsigned dimension 3; /* hourly remote access, secondary
FILL 4 byte dimension 12 fill tag $$; /* space for 2 more access types
PRIMEDAYS structure byte unsigned; /* bits representing primary days
    MONDAY bitfield; /* bit clear means this is a primary day
    TUESDAY bitfield; /* bit set means this is an off day
    WEDNESDAY bitfield;
    THURSDAY bitfield;
    FRIDAY bitfield;
    SATURDAY bitfield;
    SUNDAY bitfield;
end PRIMEDAYS;
FILL_5 byte dimension 1 fill tag $$;

PRI byte unsigned; /* base process priority
QUEPRI byte unsigned; /* maximum job queuing priority
MAXJOBS word unsigned; /* maximum jobs for UIC allowed
/* 0 means no limit
MAXACCTJOBS word unsigned; /* maximum jobs for account allowed
/* 0 means no limit

```

```
MAXDETACH word unsigned; /* maximum detached processes for UIC
/* 0 means no limit
PRCNT word unsigned; /* subprocess creation limit
BIOLM word unsigned; /* buffered I/O limit
DIOLM word unsigned; /* direct I/O limit
TQCNT word unsigned; /* timer queue entry limit
ASTLM word unsigned; /* AST queue limit
ENQLM word unsigned; /* enqueue limit
FILLM word unsigned; /* open file limit
SHRFILLM word unsigned; /* shared file limit
WSQUOTA longword unsigned; /* working set size quota
DFWSCNT longword unsigned; /* default working set size
WSEXTENT longword unsigned; /* working set size limit
PGFLQUOTA longword unsigned; /* page file quota
CPUTIM longword unsigned; /* CPU time quota
BYTLM longword unsigned; /* buffered I/O byte count limit
PBYTLM longword unsigned; /* paged buffer I/O byte count limit
JTQUOTA longword unsigned; /* job-wide logical name table creation quota
PROXY LIM word unsigned; /* number of proxies user can grant
PROXIES word unsigned; /* number of proxies granted
ACCOUNT LIM word unsigned; /* number of sub-accounts allowed
ACCOUNTS word unsigned; /* number of sub-accounts in use
FILL 99 byte dimension 64 fill tag $$;
constant FIXED equals . prefix UAF$ tag K; /* length of fixed portion
constant FIXED equals . prefix UAF$ tag C; /* length of fixed portion
FILL 100 byte dimension 68 fill tag $$; /* user-extensible area
constant "LENGTH" equals . prefix UAF$ tag K;
constant "LENGTH" equals . prefix UAF$ tag C;
end UAFDEF;
end_module $UAFDEF;
```

```
module $UASDEF;
```

```
/*
```

```
/* UNIBUS ADDRESS SPACE REGISTER DEFINITIONS FOR DW750
```

```
/* (SECOND UNIBUS ADAPTER ON 11/750)
```

```
/*
```

```
aggregate UASDEF structure prefix UASS:
```

```
    FILL_1 byte dimension 5216 fill prefix UASDEF tag $$;
```

```
    IP structure;
```

```
/* INTER-PROCESSOR EXERCISER COMMUNICATOR
```

```
        FILL_2 word dimension 2 fill prefix UASDEF tag $$; /* ADDRESS AND DATA REGISTERS NOT CURRENTLY USED
```

```
        CR1_OVERLAY union fill;
```

```
            IP CR1 word unsigned;
```

```
/* THE THIRD IPEC REGISTER, CR1
```

```
            CRT_BITS structure fill;
```

```
                FILL_3 bitfield length 12 fill prefix UASDEF tag $$; /* SKIP BITS OF NO INTEREST
```

```
                IP CR1_PIE bitfield mask; /* POWERFAIL INTERRUPT ENABLE
```

```
                IP CR1_PDN bitfield mask; /* POWER DOWN STATUS BIT
```

```
            end CR1_BITS;
```

```
        end CR1_OVERLAY;
```

```
    end IP;
```

```
end UASDEF;
```

```
end_module $UASDEF;
```

module \$UBADEF;

/*

/* UNIBUS ADAPTER REGISTER OFFSET DEFINITIONS

/*

aggregate UBADEF structure prefix UBAS;

CSR_OVERLAY union fill;

CSR_longword unsigned;

CSR_BITS structure fill;

CSR_ADCOD bitfield length 8;

FILE_1 bitfield length 8 fill prefix UBADEF

CSR_UBIC bitfield mask;

CSR_UBPDN bitfield mask;

CSR_UBIIP bitfield mask;

FILE_2 bitfield length 2 fill prefix UBADEF

CSR_OT bitfield mask;

CSR_PU bitfield mask;

CSR_PD bitfield mask;

FILE_3 bitfield length 2 fill prefix UBADEF

CSR_XMFLT bitfield mask;

CSR_MT bitfield mask;

CSR_IS bitfield mask;

CSR_URD bitfield mask;

CSR_WS bitfield mask;

CSR_PE bitfield mask;

end CSR_BITS;

end CSR_OVERLAY;

CR_OVERLAY union fill;

CR_longword unsigned;

CR_BITS structure fill;

CR_INIT bitfield mask;

CR_UBPF bitfield mask;

CR_CNFIG bitfield mask;

CR_SUEFIE bitfield mask;

CR_USEFIE bitfield mask;

CR_BRIE bitfield mask;

CR_IFSIE bitfield mask;

CR_ARLVL bitfield mask length 2;

FILE_4 bitfield length 17 fill prefix UBADEF

CR_MRDSB bitfield length 5;

end CR_BITS;

end CR_OVERLAY;

SR_OVERLAY union fill;

SR_longword unsigned;

SR_BITS structure fill;

SR_SSYNC bitfield mask;

SR_UBSTO bitfield mask;

SR_LER bitfield mask;

SR_MRPE bitfield mask;

SR_IVMR bitfield mask;

SR_DPPE bitfield mask;

SR_CXTMO bitfield mask;

SR_CXTER bitfield mask;

SR_CRD bitfield mask;

/*CONFIGURATION STATUS REGISTER

/* ADAPTER CODE FIELD

tag \$\$; /* RESERVED BITS

/* UNIBUS INITIALIZATION COMPLETE

/* UNIBUS POWER DOWN

/* UNIBUS INITIALIZATION IN PROGRESS

tag \$\$; /* RESERVED BITS

/* OVER TEMPERATURE

/* ADAPTER POWER UP

/* ADAPTER POWER DOWN

tag \$\$; /* RESERVED BITS

/* TRANSMITTER FAULT

/* MULTIPLE TRANSMITTERS

/* INTERLOCK SEQUENCE FAULT

/* UNEXPECTED READ DATA

/* WRITE SEQUENCE DATA

/* SBI PARITY ERROR

/*CONTROL REGISTER

/* ADAPTER INITIALIZATION

/* UNIBUS POWER FAIL

/* CONFIGURATION INTERRUPT ENABLE

/* SBI TO UNIBUS ERROR FIELD INTERRUPT ENABLE

/* UNIBUS TO SBI ERROR FIELD INTERRUPT ENABLE

/* BUS REQUEST INTERRUPT ENABLE

/* INTERRUPT FIELD SWITCH INTERRUPT ENABLE

/* ADAPTER REQUEST LEVEL

tag \$\$; /* RESERVED BITS

/* MAP REGISTER DISABLE

/*STATUS REGISTER

/* UNIBUS SLAVE SYNC TIMEOUT

/* UNIBUS SELECT TIMEOUT

/* LOST ERROR

/* MAP REGISTER PARITY ERROR

/* INVALID MAP REGISTER

/* DATAPATH PARITY ERROR

/* COMMAND TRANSMISSION TIMEOUT

/* COMMAND TRANSMISSION ERROR

/* CORRECTED READ DATA

end

/*

/*

/*

agg

```

SR_RDS bitfield mask; /* READ DATA SUBSTITUTE
SR_RDT0 bitfield mask; /* READ DATA TIMEOUT
SR_BRID bitfield mask; /* BUS REQUEST INTERRUPT DONE
FILL 5 bitfield length 12 fill prefix UBADEF tag $$; /* RESERVED BITS
SR_BRRVF bitfield length 4; /* BUS REQUEST RECEIVE VECTOR FULL
SR_BRSVF bitfield mask; /* BUS REQUEST SEND VECTOR FULL
SR_RIE bitfield mask; /* REQUEST INTERRUPT ENABLED
SR_UNIFS bitfield mask; /* UNIBUS INTERRUPT FIELD SWITCH
end SR_BITS;
end SR_OVERLAY;
DCR longword unsigned; /*DIAGNOSTIC CONTROL REGISTER
FMER OVERLAY union fill; /*FAILED MAP ENTRY REGISTER
FMER longword unsigned; /* FAILED MAP REGISTER NUMBER
FMER BITS structure fill;
FMER_MRN bitfield length 9;
end FMER_BITS;
end FMER_OVERLAY;
FUBAR OVERLAY union fill; /*FAILED UNIBUS ADDRESS REGISTER
FOBAR longword unsigned; /* FAILED SBI TO UNIBUS ADDRESS
FUBAR BITS structure fill;
FOBAR_ADR bitfield length 18;
end FUBAR_BITS;
end FUBAR_OVERLAY;
FILL 6 longword dimension 2 fill prefix UBADEF tag $$; /*RESERVED REGISTERS
BRSVR longword unsigned dimension 4; /*BUS REQUEST SEND VECTOR REGISTERS
BRRVR OVERLAY union fill; /*BUS REQUEST RECEIVE VECTOR REGISTER
BRPVR longword unsigned dimension 4;
BRRVR BITS structure fill; /* INTERRUPT VECTOR ADDRESS
BRRVR_IVA bitfield length 16;
FILL 7 bitfield length 15 fill prefix UBADEF tag $$; /* RESERVED BITS
BRRVR_AIR bitfield mask; /* ADAPTER INTERRUPT REQUEST PENDING
end BRRVR_BITS;
end BRRVR_OVERLAY;
DPR_OVERLAY union fill; /*DATAPATH REGISTERS
DPR longword unsigned dimension 16;
DPR_BITS structure fill; /* BUFFERED UNIBUS ADDRESS
DPR_ADDR bitfield length 16; /* BUFFER STATE FLAGS
DPR_STATE bitfield length 8; /* RESERVED BITS
FILL 8 bitfield length 5 fill prefix UBADEF tag $$; /* DATAPATH FUNCTION
DPR_DPF bitfield mask; /* BUFFER TRANSFER ERROR
DPR_XMTER bitfield mask; /* BUFFER NOT EMPTY
DPR_BNE bitfield mask;
end DPR_BITS;
end DPR_OVERLAY;
FILL 9 byte dimension 1920 fill prefix UBADEF tag $$; /* VALUE IS 2048-128
MAP_OVERLAY union fill; /*MAP REGISTERS
MAP longword unsigned dimension 496;
MAP_BITS structure fill; /* SBI PAGE ADDRESS
MAP_ADDR bitfield length 21; /* DATAPATH DESIGNATOR
MAP_DPD bitfield length 4; /* BYTE OFFSET
MAP_BO bitfield mask; /* RESERVED BITS
FILL 10 bitfield length 5 fill prefix UBADEF tag $$; /* MAP REGISTER VALID
MAP_VALID bitfield mask;
end MAP_BITS;
constant MAXDP equals 15 prefix UBA tag $C; /*MAXIMUM DATAPATH !

```

SYSDEFQZ.SDL:1

16-SEP-1984 16:45:41.35^{0.14} Page 60

```

end MAP_OVERLAY;
end UBADEF;

end_module SUBADEF;

```

SYS

MOD

10

15



10

15

10

10

10

agg

1

10

10



10

10



```
module SUBIDEF;
```

```
/*  
/* UNIBUS INTERCONNECT (VAX 11/750 & 11/730) REGISTER OFFSETS AND FIELDS  
/*-
```

```
aggregate UBIDEF union prefix UBI$;  
    DPR longword unsigned dimension 4;
```

```
/*DATAPATH REGISTERS  
/* (DPR 0 NOT IMPLEMENTED)
```

```
    DPR_BITS structure fill;
```

```
        DPR_PUR bitfield mask;
```

```
        FILC_1 bitfield length 28 fill prefix UBIDEF tag $$; /* RESERVED BITS
```

```
        DPR_OCE bitfield mask;
```

```
        DPR_NXM bitfield mask;
```

```
        DPR_ERROR bitfield mask;
```

```
/* DATAPATH PURGE  
/* UNCORRECTABLE ERROR  
/* NON-EXISTENT MEMORY  
/* ERROR (UCE!NXM)
```

```
    end DPR_BITS;
```

```
end UBIDEF;
```

```
aggregate UBIDEF1 structure prefix UBI$;
```

```
    FILL 6 byte dimension 16 fill prefix UBIDEF tag $$;
```

```
    DSR_OVERLAY union fill;
```

```
        DSR longword unsigned dimension 4;
```

```
/*DIAGNOSTIC STATUS REGISTERS  
/* (DSR 0 NOT IMPLEMENTED)
```

```
        DSR_BITS structure fill;
```

```
            FILL 2 bitfield length 27 fill prefix UBIDEF tag $$; /* RESERVED BITS
```

```
            DSR_CD bitfield mask;
```

```
            DSR_BF bitfield length 4;
```

```
/* ALL 4 BYTES IN BDP FULL  
/* BYTE 0,1,2,3 IN BDP HAS VALID DATA
```

```
        end DSR_BITS;
```

```
    end DSR_OVERLAY;
```

```
end UBIDEF1;
```

```
aggregate UBIDEF2 structure prefix UBI$;
```

```
    FILL 7 byte dimension 16 fill prefix UBIDEF tag $$;
```

```
    SR_OVERLAY union fill;
```

```
        SR longword unsigned;
```

```
        SR_BITS structure fill;
```

```
            FILL 3 bitfield length 14 fill prefix UBIDEF tag $$; /* RESERVED BITS
```

```
            SR_UDE bitfield mask;
```

```
            SR_MRPE bitfield mask;
```

```
            SR_NXM bitfield mask;
```

```
            FILC_4 bitfield length 14 fill prefix UBIDEF tag $$; /* RESERVED BITS
```

```
            SR_UCE bitfield mask;
```

```
/*UB STATUS REGISTER:
```

```
/* UNCORRECTED WRITE ERROR  
/* MAP REGISTER PARITY ERROR  
/* NONEXISTENT MEMORY REF  
/* UNCORRECTED READ ERROR
```

```
        end SR_BITS;
```

```
/*END OF CPU_SPECIFIC REGISTERS
```

```
    end SR_OVERLAY;
```

```
end UBIDEF2;
```

```
aggregate UBIDEF3 structure prefix UBI$;
```

```
    FILL 5 byte dimension 2048 fill prefix UBIDEF tag $$; /*RESERVE ^X800 BYTES
```

```
    MAP Longword unsigned dimension 496;
```

```
    constant MAXDP equals 3 prefix UBI tag $C;
```

```
    constant PURCNT equals 10 prefix UBI tag $C;
```

```
/*MAP REGISTERS, SAME FORMAT AS UBA
```

```
/*MAXIMUM DATAPATH !
```

```
/*MAX ! OF TESTS OF PURGE DONE
```

```
end UBIDEF3;
```

```
end_module SUBIDEF;
```

SYSDEFQZ.SDL;1

16-SEP-1984 16:45:41.35 ^{F 14} Page 62

SYS

/*

/*

/*

/*-

/*

/*+

/*-

/*

/*

/*

/*+

/*-

/*

/*+

```
module SUBMDDEF;
```

```
/*
```

```
/* UBMD - UNIBUS Map Descriptor used to record UNIBUS map registers  
/* and datapaths allocated.  
/*
```

```
aggregate UBMDDEF structure prefix UBMD$;
```

```
MAPREG word unsigned;
```

```
NUMREG byte unsigned;
```

```
DATAPATH byte unsigned;
```

```
/* Starting map register
```

```
/* Number of registers in extent
```

```
/* Associated Buffered datapath
```

```
end UBMDDEF;
```

```
end_module SUBMDDEF;
```

```
{
  * UCB - UNIT CONTROL BLOCK
  * THERE IS ONE UCB FOR EACH DEVICE UNIT IN A SYSTEM.
}
```

```
module $UCBDEF;
```

```
aggregate UCBDEF structure prefix UCBS:
```

```
  FQFL OVERLAY union fill;
```

```
    FQFL longword unsigned;
    UNIT SEED word unsigned;
    MB SEED word unsigned;
    RQFL longword unsigned;
```

```
  end FQFL OVERLAY;
```

```
  FQBL OVERLAY union fill;
```

```
    FQBL longword unsigned;
    RQBL longword unsigned;
```

```
  end FQBL OVERLAY;
```

```
  SIZE word unsigned;
```

```
  TYPE byte unsigned;
```

```
  FIPL byte unsigned;
```

```
  FPC OVERLAY union fill;
```

```
    FPC longword unsigned;
    ASTQFL longword unsigned;
    PARTNER character;
```

```
  end FPC OVERLAY;
```

```
  FR3 OVERLAY union fill;
```

```
    FR3 longword unsigned;
    ASTQBL longword unsigned;
```

```
  end FR3 OVERLAY;
```

```
  FR4 OVERLAY union fill;
```

```
    FR4 longword unsigned;
    MB_FR4_FIELDS structure fill;
```

```
      MSGMAX word unsigned;
      MSGCNT word unsigned;
```

```
    end MB_FR4_FIELDS;
```

```
    FIRST longword unsigned;
```

```
  end FR4 OVERLAY;
```

```
  BUFQUO OVERLAY union fill;
```

```
    BUFQUO word unsigned;
    DSTADDR word unsigned;
```

```
  end BUFQUO OVERLAY;
```

```
  SRCADDR word unsigned;
```

```
  ORB longword unsigned;
```

```
  LOCKID OVERLAY union fill;
```

```
    LOCKID longword unsigned;
    CPID longword unsigned;
```

```
  end LOCKID OVERLAY;
```

```
  CRB longword unsigned;
```

```
  DDB longword unsigned;
```

```
  PID longword unsigned;
```

```
  LINK longword unsigned;
```

```
  VCB longword unsigned;
```

```
  DEVCHAR union fill;
```

```
    DEVCHAR quadword unsigned;
```

```
/*FORK QUEUE FORWARD LINK
/* UNIT NUMBER SEED
/* MB -- UNIT NUMBER SEED
/* NET -- RCV QUEUE FORWARD LINK
```

```
/*FORK QUEUE BACKWARD LINK
/* NET -- RCV QUEUE BACKWARD LINK
```

```
/*SIZE OF UCB IN BYTES
/*STRUCTURE TYPE FOR UCB
/*FORK INTERRUPT PRIORITY LEVEL
```

```
/*FORK PC
/* MB -- AST QUEUE LISTHEAD FORWARD LINK
/* NET -- PARTNER'S NODENAME
```

```
/*FORK R3
/* MB -- AST QUEUE LISTHEAD BACKWARD LINK
```

```
/*FORK R4
```

```
/* MB -- MAXIMUM MESSAGES ALLOWED
/* MB -- CURRENT NUMBER OF MESSAGES
```

```
/* NET -- ADDR OF 1ST SEG OF CHAINED MSG
```

```
/* BUFFERED I/O QUOTA CHARGED FOR THIS UCB
/* NET -- REMOTE CONNECT NO.
```

```
/* NET -- LOCAL CONNECT NO.
/* OBJECT'S RIGHTS BLOCK ADDRESS
```

```
/*DEVICE LOCK ID
/*PID CHARGED FOR BUFQUO BY UCBCREDEL
```

```
/*ADDRESS OF PRIMARY CHANNEL REQUEST BLOCK
/*BACKPOINTER TO DEVICE DATA BLOCK
/*PROCESS ID OF OWNER PROCESS
/*ADDRESS OF NEXT UCB FOR RESPECTIVE DDB
/*ADDRESS OF VOLUME CONTROL BLOCK
/*DEVICE CHARACTERISTIC BITS
/* Device characteristic bits quadword
```

```

DEVCHAR_Q_BLOCK structure fill;
  DEVCHAR longword unsigned;
  DEVCHAR2 longword unsigned;
end DEVCHAR_Q_BLOCK;
end DEVCHAR;
DEVCLASS byte unsigned;
DEVTYPE byte unsigned;
DEVBUFSIZ word unsigned;
DEVDEPEND_Q_OVERLAY union fill;
  DEVDEPEND quadword unsigned;
  DEVDEPEND_Q_BLOCK structure;
    DEVDEPEND_OVERLAY union fill;
      DEVDEPEND longword unsigned;
      DISK_DEVDEPEND structure;
        SECTORS byte unsigned;
        TRACKS byte unsigned;
        CYLINDERS word unsigned;
      end DISK_DEVDEPEND;
      TERM_DEVDEPEND structure;
        TERM_DEVDEPEND_FILL byte dimension 3
          fill prefix UCBDEF tag $$;
        VERTSZ byte unsigned;
      end TERM_DEVDEPEND;
      NET_DEVDEPEND structure;
        LOCSRV byte unsigned;
        REMSRV byte unsigned;
        BYTESTOGO word unsigned;
      end NET_DEVDEPEND;
      JNL_SEQNO longword unsigned;
    end DEVDEPEND_OVERLAY;
    DEVDEPN2 OVERLAY union fill;
      DEVDEPN2 longword unsigned;
      TT_DEVP1 longword unsigned;
    end DEVDEPN2 OVERLAY;
  end DEVDEPEND_Q_BLOCK;
end DEVDEPEND_Q_OVERLAY;
IOQFL longword unsigned;
IOQBL longword unsigned;
UNIT word unsigned;
CHARGE_OVERLAY union fill;
  CHARGE word unsigned;
  RWAITCNT word unsigned;
  CTRLR_ALLOC_FIELDS structure fill;
    CM1 byte unsigned;
    CM2 byte unsigned;
  end CTRLR_ALLOC_FIELDS;
end CHARGE_OVERLAY;
IRP longword unsigned;
REFC word unsigned;
DIPL_OVERLAY union fill;
  DIPL byte unsigned;
  STATE byte unsigned;
end DIPL_OVERLAY;
AMOD byte unsigned;
AMB longword unsigned;
STS_OVERLAY union fill;

```

```

/* Original device characteristic bits
/* Extended device characteristic bits

```

```

/*DEVICE CLASS
/*DEVICE TYPE
/*DEVICE DEFAULT BUFFER SIZE
/*DEVICE DEPENDENT DATA
/*Device dependent quadword

```

```

/* First device dependent longword
/* Disk fields
/* Sectors per track
/* Track per cylinder
/* Cylinders per disk

```

```

/* Terminal fields

```

```

/* Vertical page size (lines per page)

```

```

/* Network fields
/* Local link services
/* Remote link services
/* No. of bytes left in rcv bfr

```

```

/* Journal -- Running sequence number

```

```

/* Second device dependent long word
/* Terminal -- Device dependent long word

```

```

/*I/O QUEUE LISTHEAD FORWARD LINK
/*I/O QUEUE LISTHEAD BACKWARD LINK
/*PHYSICAL DEVICE UNIT NUMBER

```

```

/*MAILBOX BYTE COUNT QUOTA CHARGE
/* CLASS DRIVERS -- THREADS WAITING RESOURCES

```

```

/* LEVEL 1 CONTROLLER ALLOCATION MASK
/* LEVEL 2 CONTROLLER ALLOCATION MASK

```

```

/*CURRENT I/O REQUEST PACKET ADDRESS
/*REFERENCE COUNT OF PROCESSES

```

```

/*DEVICE INTERRUPT PRIORITY LEVEL
/* NET -- LINK STATE FOR NETWORK TRANSITIONS

```

```

/*ALLOCATION ACCESS MODE
/*ASSOCIATED UNIT CONTROL BLOCK POINTER

```

```

STS longword unsigned;
STS word unsigned;
STS_BITS structure fill;
  TIM bitfield mask;
  INT bitfield mask;
  ERLOGIP bitfield mask;
  CANCEL bitfield mask;
  ONLINE bitfield mask;
  POWER bitfield mask;
  TIMEOUT bitfield mask;
  INTTYPE bitfield mask;
  BSY bitfield mask;
  MOUNTING bitfield mask;
  DEADMO bitfield mask;
  VALID bitfield mask;
  UNLOAD bitfield mask;
  TEMPLATE bitfield mask;

  MNTVERIP bitfield mask;
  WRONGVOL bitfield mask;
  DELETEUCB bitfield mask;
  LCL_VALID bitfield mask;
  SUPRVMSG bitfield mask;

  MNTVERPND bitfield mask;
  DISMOUNT bitfield mask;
  CLUTRAN bitfield mask;
end STS_BITS;
end STS_OVERLAY;
DEVSTS_OVERLAY union fill;
DEVSTS word unsigned;
DEVSTS_GENERAL_BITS structure fill;
  JOB bitfield mask;
  devsts_gen_fill bitfield length 5 fill;
  TEMPL_BSY bitfield mask;
end DEVSTS_GENERAL_BITS;
DEVSTS_MAILBOX_BITS structure fill;
  PRMBX bitfield mask;
  DELMBX bitfield mask;
  devsts_mb_fill bitfield length 1 fill;
  SHMMBX bitfield mask;
end DEVSTS_MAILBOX_BITS;
DEVSTS_TERM_BITS structure fill;
  devsts_Tt_fill bitfield length 1 fill;
  TT_TIMO bitfield mask;
  TT_NOTIF bitfield mask;
  TT_HANGUP bitfield mask;
  TT_DEVSTS_FILL bitfield length 15-^;
  TT_NOLOGINS bitfield mask;
end DEVSTS_TERM_BITS;
DEVSTS_NET_BITS structure fill;
  devsts_net_fill1 bitfield length 2 fill;
  NT_BFROVF bitfield mask;
  devsts_net_fill2 bitfield fill;
  NT_NAME bitfield mask;

```

/*DEVICE UNIT STATUS

```

/* TIME OUT ENABLED (1=YES)
/* INTERRUPT EXPECTED (1=YES)
/* ERROR LOG IN PROGRESS ON UNIT (1=YES)
/* CANCEL I/O ON UNIT (1=YES)
/* UNIT ONLINE (1=YES)
/* POWER FAILED WHILE UNIT BUSY (1=YES)
/* UNIT TIMED OUT (1=YES)
/* RECEIVER INTERRUPT IF SET
/* UNIT IS BUSY (1=YES)
/* DEVICE IS BEING MOUNTED
/* DEALLOCATE AT DISMOUNT
/* VOLUME IS SOFTWARE VALID
/* UNLOAD VOLUME AT DISMOUNT
/* SET IF THIS IS TEMPLATE UCB
{ FROM WHICH OTHER UCB'S FOR
{ THIS DEVICE TYPE ARE MADE
/* MOUNT VERIFICATION IN PROGRESS
/* WRONG VOLUME DETECTED DURING MOUNT VERIFICATION
/* DELETE THIS UCB WHEN REFC REACHES ZERO
/* VOLUME IS VALID ON THE LOCAL NODE
/* IF SET, SUPPRESS SUCCESS TYPE MOUNT VER. MSGS.
{ CLEARED BY ANY ERROR IN MOUNT VERIFICATION
/* MOUNT VERIFICATION IS PENDING ON BUSY DEVICE.
/* DISMOUNT IN PROGRESS
/* VAXcluster STATE TRANSITION IN PROGRESS

```

/*DEVICE DEPENDENT STATUS

```

/* Generally used bits
/* Job Controller notified

/* Template UCB is busy

/* Mailbox status bits
/* Permanent mailbox
/* Mailbox marked for delete

/* Shared memory mailbox

/* Terminal status bits

/* Terminal read timeout in progress
/* Terminal user notified of unsolicited data
/* Process hang up
/* fill to the end the word
/* NOLOGINS ALLOWED

/* Network status bits

/* Too many bytes rcvd

/* Link has declared a connect name

```

```

    NT_BREAK bitfield mask;
end DEVSTS_NET_BITS;
DEVSTS_DISKS structure fill;

    ECC bitfield mask;
    DIAGBUF bitfield mask;
    NOCNVRT bitfield mask;
    DX_WRITE bitfield mask;
    DATACACHE bitfield mask;
end DEVSTS_DISKS;
DEVSTS_MSCP_CLASS_BITS structure fill;
    byte_fill bitfield length 8 fill;
    MSCP_MNTVERIP bitfield mask;
    MSCP_INITING bitfield mask;
    MSCP_WAITBMP bitfield mask;
    MSCP_FLOVR bitfield mask;
    MSCP_PKACK bitfield mask;
    MSCP_WRTM bitfield mask;
end DEVSTS_MSCP_CLASS_BITS;
DEVSTS_TAPE_CLASS_BITS structure fill;

    TU_OVRSQCHK bitfield mask;
    TU_TRACEACT bitfield mask;
    TU_SEQNOP bitfield mask;
end DEVSTS_TAPE_CLASS_BITS;
DEVSTS_JOURNAL_BITS structure fill;
    devsts_jnl_fill bitfield length 4 fill;
    PERM_JNL bitfield mask;
    KNOWN_JNL bitfield mask;
    JNL_CES bitfield mask;
    JNL_SLV bitfield mask;
    CDELE_PND bitfield mask;
    JNL_URMAST bitfield mask;
end DEVSTS_JOURNAL_BITS;
end DEVSTS_OVERLAY;
OLEN word unsigned;
DUETIM longword unsigned;
OPCNT longword unsigned;
SVPN_OVERLAY union fill;
    SVPN longword unsigned;
    LOGADR longword unsigned;
end SVPN_OVERLAY;
SVAPTE longword unsigned;
BOFF word unsigned;
BCNT word unsigned;
ERTCNT byte unsigned;
ERTMAX byte unsigned;
ERRCNT word unsigned;
PDT_OVERLAY union fill;
    PDT longword unsigned;
    JNL_MCSID longword unsigned;
end PDT_OVERLAY;
DDT longword unsigned;
MEDIA_ID_OVERLAY union fill;

```

```

/* Link is being broken
/* Disk (all disks) status bits
{ (using the low order byte, for
{ compatibility with class driver
{ usage of the high order word)
/* ECC correction was made
/* Diagnostic buffer specified
/* No LBN to media address conversion
/* Console floppy write operation
/* Data blocks being cached
/* MSCP class driver bits
{ (using the high order byte only)
/* Mount verification in progress
/* UCB is being initialized
/* RWAITCNT has been bumped
/* Bit toggled everytime a failover succeeds.
/* Set when a IOS_PACKACK is in progress.
/* Unit MSCP write protected in some way.
/* Tape class driver bits
{ (using the low order byte only)
/* Override sequence checking
/* IRP trace table active
/* Sequential NOP tape operation in progress
/* Journal status bits
/* Permanent journal device
/* Known journal
/* Cluster journal
/* Slave journal UCB
/* Cluster delete operation pending
/* Device unmastered
/* Device queue length
/*DUE TIME FOR I/O COMPLETION
/*COUNT OF OPERATIONS COMPLETED
/*SYSTEM VIRTUAL PAGE/MAP REGISTER NUMBER
/* MB -- LOGICAL NAME BLOCK ADDRESS
/*SYSTEM VIRTUAL ADDRESS OF PTE
/*BYTE OFFSET IN FIRST PAGE
/*BYTE COUNT OF TRANSFER
/*ERROR LOG DEVICE CURRENT ERROR RETRY COUNT
/*ERROR LOG DEVICE MAXIMUM ERROR RETRY COUNT
/*DEVICE ERROR COUNT
/*ADDR OF PORT DESCRIPTOR TABLE
/*MASTER NODE'S CSID (JOURNALING)
/*ADDR OF DDT (OPTIONAL BUT PREFERRED)

```

```

MEDIA_ID longword unsigned;
MEDIA_ID_SUBFIELDS structure fill;
  MEDIA_ID_NN bitfield length 7;
  MEDIA_ID_N2 bitfield length 5;
  MEDIA_ID_N1 bitfield length 5;
  MEDIA_ID_N0 bitfield length 5;
  MEDIA_ID_T1 bitfield length 5;
  MEDIA_ID_T0 bitfield length 5;
end MEDIA_ID_SUBFIELDS;
end MEDIA_ID_OVERLAY;
constant "LENGTH" equals . prefix UCBS tag K;
constant "LENGTH" equals . prefix UCBS tag C;
#UCB_LENGTH = .; /* for $TTYUCBDEF
end UCBDEF;

/*
/* DEVICE DEPENDENT UCB EXTENSIONS
/*
/* MAILBOX
/*

aggregate UCBDEF3 structure prefix UCBS;
  FILL 7 byte dimension UCBSK_LENGTH fill prefix UCBDEF tag $$;
  MB_WAST longword unsigned; /*WRITE ATTENTION AST BLOCK ADDR
  MB_RAST longword unsigned; /*READ ATTENTION AST BLOCK ADDR
  MB_MBX longword unsigned; /*MAILBOX CONTROL BLOCK ADDR
  MB_SHB longword unsigned; /*SHARED MEM. CONTROL BLOCK ADDR
  MB_WIOQFL longword unsigned; /*WRITE I/O QUEUE FORWARD LINK
  MB_WIOQBL longword unsigned; /*WRITE I/O QUEUE BACKWARD LINK
  MB_PORT longword unsigned; /*SHARED MEM. PORT NUMBER
  constant MB_LENGTH equals . prefix UCBS tag K; /*SIZE OF MAILBOX UCB
  constant MB_LENGTH equals . prefix UCBS tag C; /*SIZE OF MAILBOX UCB
end UCBDEF3;

/*
/* ERROR LOG DEVICES (ALL)
/*

aggregate UCBDEF4 structure prefix UCBS;
  FILL 8 byte dimension UCBSK_LENGTH fill prefix UCBDEF tag $$;
  SLAVE byte unsigned; /*SLAVE CONTROLLER NUMBER
  SPR byte unsigned; /*SPARE UNUSED BYTE
  FEX byte unsigned; /*FUNCTION DISPATCH TABLE INDEX
  CEX byte unsigned; /*CASE TABLE FUNCTION EXECUTION INDEX
  EMB longword unsigned; /*ADDRESS OF ERROR MESSAGE BUFFER
  FILL_1 word fill prefix UCBDEF tag $$; /*SPARE UNUSED WORD
  FUNC word unsigned; /*I/O FUNCTION MODIFIERS
  DPC longword unsigned; /*SAVED DRIVER SUBROUTINE RETURN ADDRESS
  constant ERL_LENGTH equals .; /*SIZE OF ERROR LOG UCB
  constant ERL_LENGTH equals . tag C; /*SIZE OF ERROR LOG UCB
end UCBDEF4;

/*
/* DUAL PORTED DEVICES (ALL DISKS AND MOST TAPES)
/*

```

```

aggregate DUALPATH_EXTENSION structure prefix UCBS;
  fill dualpath byte dimension UCBSK_ERL_LENGTH fill;
  DUAL_PATH union fill;
    OLD_DUAL_PATH structure fill;
      DP_DDB longword unsigned;
      DP_LINK longword unsigned;
      DP_ALTUCB longword unsigned;
    end OLD_DUAL_PATH;
    PREFERED_DUAL_PATH structure fill;
      "2P_DDB" longword unsigned;
      "2P_LINK" longword unsigned;
      "2P_ALTUCB" longword unsigned;
    end PREFERED_DUAL_PATH;
  end DUAL_PATH;
  constant DP_LENGTH equals .;
  constant DP_LENGTH equals . tag C;
  constant "2P_LENGTH" equals .;
  constant "2P_LENGTH" equals . tag C;
  #2P_LENGTH = .;
end DUALPATH_EXTENSION;

/*
/* ALL DISKS AND TAPES
/*

aggregate DISKTAPE_UCB_EXTENSION structure prefix UCBS;
  fill disktape byte dimension #2P_LENGTH fill;
  DIRSEQ structure word unsigned;
  filler bitfield length 15 fill;
  AST_ARMED bitfield mask;
end DIRSEQ;
ONLCNT byte unsigned;
reserved byte fill;
DISKTAPE_OVERLAY union fill;
  MAXBLOCK longword unsigned;
  RECORD longword unsigned;
end DISKTAPE_OVERLAY;
constant LCL_TAPE_LENGTH equals .;
constant LCL_TAPE_LENGTH equals . tag C;
MAXBCNT longword unsigned;
DCCB longword unsigned;
#BEGIN_LOCAL_DISKS = .;
#BEGIN_MSCP = .;
end DISKTAPE_UCB_EXTENSION;

aggregate LCL_DISK_UCB_EXTENSION structure prefix UCBS;
  fill lcl_disk byte dimension #BEGIN_LOCAL_DISKS fill;
  MEDIA structure longword unsigned;
  DA word unsigned;
  DC word unsigned;
end MEDIA;
BCR structure longword unsigned;
BCR word unsigned;
end BCR;
EC1 word unsigned;
EC2 word unsigned;

```

{ Setup two versions of the dual-path names
 { This is the old way of naming things
 /* Pointer to alternate DDB
 /* Address of next UCB for this DDB
 /* Addr of alternate UCB for this unit
 { This is the preferred way of naming things
 /* Pointer to alternate DDB
 /* Address of next UCB for this DDB
 /* Addr of alternate UCB for this unit
 /* Size of dual path UCB
 /* size of dual path UCB
 /* Size of dual path UCB
 /* size of dual path UCB
 /* Directory sequence number
 { skip value portion of sequence number
 /* Blocking AST armed flag
 /* Online count
 { Reserved byte
 /* Random access device highest block
 /* Current tape position or frame counter
 /* Size of local tape UCB
 /* Size of local tape UCB
 /* Maximum transfer BCNT
 /* Pointer to data cache control block
 /* Media address (longword)
 /* Saved desired sector/track address register
 /* Saved desired cylinder address register
 /* Byte count register
 /* ECC position register
 /* ECC pattern register

```

OFFSET word unsigned;          /* Current offset register contents
OFFNDX byte unsigned;          /* Current offset table index
OFFRTC byte unsigned;          /* Current offset retry count
constant LCL_DISK_LENGTH equals :; /* Size of local disk UCB
constant LCL_DISK_LENGTH equals : tag C; /* Size of local disk UCB

/*
/* FLOPPY DEPENDENT BIT DEFINITIONS
/*
DX_BUF longword unsigned;      /* ADDRESS OF SECTOR BUFFER
DX_BFPNT longword unsigned;    /* CURRENT SECTOR BUFFER POINTER
DX_RXDB longword unsigned;     /* SAVED RECEIVER DATA BUFFER
DX_BCR word unsigned;          /* CURRENT FLOPPY BYTE COUNT
DX_SCTCNT byte unsigned;       /* CURRENT SECTOR BYTE COUNT
FICL_2 byte fill prefix UCBDEF tag $$; /* SPARE UNUSED BYTE

end LCL_DISK_UCB_EXTENSION;
/*
/* MSCP DISKS AND TAPES UCB EXTENSION
/*
aggregate MSCP_UCB_EXTENSION structure prefix UCB$;
mscp_fill byte dimension #BEGIN_MSCP fill;
CDDB longword unsigned;        /* Pointer to active CDDB
"2P CDDB" longword unsigned;    /* Pointer to alternate CDDB
CDDB_LINK longword unsigned;    /* Pointer to next UCB in CDDB chain
CDT longword unsigned;          /* Pointer to active CDT
UNIT_ID quadword unsigned;      /* Unique MSCP unit identifier
MSCPUNIT word unsigned;         /* Primary path MSCP unit number
"2P MSCPUNIT" word unsigned;    /* Secondary path MSCP unit number
MSCPDEVPARAM longword unsigned; /* MSCP device-dependent parameters
WAIT_CDDB longword unsigned;    /* Address of the CDDB waiting for mnt. ver.
/* to complete on this UCB
UNIT_FLAGS word unsigned;       /* MSCP unit flags
reserved word unsigned fill;    /* reserved word, for alignment
MSCP_RESV quadword unsigned;    /* Reserved for MSCP enhancements
constant MSCP_DISK_LENGTH equals :; /* Size of MSCP disk UCB
constant MSCP_TAPE_LENGTH equals :; /* Size of MSCP tape UCB

end MSCP_UCB_EXTENSION;
/*
/* NETWORK LOGICAL LINK (NETWORK MAILBOX) EXTENSION
/*
aggregate UCBDEF7 structure prefix UCB$;
FILL 11 byte dimension UCB$K_LENGTH fill prefix UCBDEF tag $$;
NT_DATSSB longword unsigned;    /* ADDRESS OF DATA SUBCHANNEL STATUS BLOCK
NT_INTSSB longword unsigned;    /* ADDRESS OF INT/LS SSB
NT_CHAN word unsigned;          /* DDCMP CHANNEL NO.
FICL 3 OVERLAY union fill;
FICL_3 word fill prefix UCBDEF tag $$; /* DUMMY FIELD
FILL 3 BITS structure fill;
LTYPE bitfield length 2;        /* LINK TYPE BITS
SEGFLD bitfield;                /* SEGMENT REQUEST COUNTS
MSGFLD bitfield;                /* MESSAGE REQUEST COUNTS

```

```

    MSGACK bitfield;
    BACKP bitfield mask;
    LNKPRI bitfield length 2;
end FILL_3_BITS;

    constant LOGLNK equals 1 prefix UCB tag $C;
end FILL_3_OVERLAY;
end UCBDEF7;

/*
/* JOURNAL DEVICE EXTENSION
/*

aggregate UCBDEF8 structure prefix UCBS;
  FILL 12 byte dimension UCBSK_ERL_LENGTH fill prefix UCBDEF tag $$;
  JNL_BCB_OVERLAY union fill;
    JNL_BCB longword unsigned;
    JNL_ADL longword unsigned;
  end JNL_BCB_OVERLAY;
  JNL_RUL longword unsigned;
  JNL_WQFL longword unsigned;
  JNL_WQBL longword unsigned;
  JNL_FQFL longword unsigned;
  JNL_FQBL longword unsigned;
  JNL_NAM character;
  JNL_NAM byte unsigned dimension 18;
  FILL 4 byte fill prefix UCBDEF tag $$;
  JNL_QUOT longword unsigned;
  JNL_ID word unsigned;
  JNL_MUNIT word unsigned;
  JNL_MASK longword unsigned;
  JNL_ASID_OVERLAY union fill;
    JNL_ASID longword unsigned;
    JNL_NDL longword unsigned;
  end JNL_ASID_OVERLAY;
  JNL_REFC longword unsigned;
  JNL_TREFC longword unsigned;
  JNL_MXENT word unsigned;
  JNL_PROT word unsigned;
  JNL_WRCNT longword unsigned;
  JNL_BWCNT longword unsigned;
  JNL_EXCNT longword unsigned;
  JNL_FAILQFL longword unsigned;
  JNL_FAILQBL longword unsigned;
  JNLSEQ_OVLY union fill;
    JNL_LSEQNO longword unsigned;
    JNL_BTSEQNO longword unsigned;
  end JNLSEQ_OVLY;
  JNL_ACBM longword unsigned;
  JNL_RMBLK longword unsigned;
  JNL_CWQFL longword unsigned;
  JNL_CWQBL longword unsigned;
  JNL_WCBFL longword unsigned;
  JNL_WCBBL longword unsigned;
  FILL 5 longword fill prefix UCBDEF tag $$;
  constant JNL_LENGTH equals . prefix UCBS tag C;

/* MESSAGE ACK/NAK
/* BACKPRESSURE (1=> NO FLOW)
/* LINK PRIORITY (IGNORED)

/* NETWORK CONSTANTS
/* CONNECT IS FOR LOGICAL LINK (NOT SINGLE MSG)

/* ADDRESS OF BUFFER CONTROL BLOCK
/* ALLOCATED JOURNAL DEVICE LIST

/* ADDRESS OF RUL (RU JOURNALS ONLY)
/* WAIT QUEUE FORWARD LINK
/* WAIT QUEUE BACKWARD LINK
/* FORCE QUEUE FORWARD LINK
/* FORCE QUEUE BACKWARD LINK
/* JOURNAL NAME LENGTH
/* JOURNAL NAME
/* SPARE
/* QUOTA FOR RU JOURNALS
/* JOURNAL ID (TAPES ONLY)
/* MASTER'S DEVICE UNIT NUMBER
/* JOURNAL MASK

/* ASSIGN ID
/* POINTER TO NAME TABLE DEVICE LIST

/* LOCAL REFERENCE COUNT
/* TOTAL REFERENCE COUNT
/* MAXIMUM ENTRY SIZE
/* PROTECTION MASK
/* WRITE COUNT
/* BUFFER WRITE COUNT
/* JOURNAL EXTEND COUNT
/* FAIL OVER WAIT Q FORWARD LINK
/* FAIL OVER WAIT Q BACKWARD LINK

/* LOCAL SEQUENCE NUMBER
/* BLOCK TRANSFER SEQ NUM (PROTO UCB ONLY)

/* ADDRESS OF ACCESS BIT MAP
/* ADDRESS OF REMASTER BLOCK
/* Cluster write Q forward link
/* Cluster write Q backward link
/* WCB LISTHEAD FORWARD LINK
/* WCB LISTHEAD BACKWARD LINK
/* SPARE
/* JOURNAL UCB LENGTH

```

end UCBDEF8;

/*
/* NI DEVICE EXTENSION
/*

aggregate UCBDEF9 structure prefix UCBS;

FILL 13 byte dimension UCBSK_LENGTH fill prefix UCBDEF tag \$\$;

NI_HQAPTR longword unsigned;

/*ADDRESS OF NI DEVICE HARDWARE ADDRESS

NI_MLTPTR longword unsigned;

/*ADDRESS OF PROTOCOL MULTICAST TABLE

constant NI_LENGTH equals : prefix UCBS tag K;

/*SIZE OF NI DEVICE UCB

constant NI_LENGTH equals : prefix UCBS tag C;

/*SIZE OF NI DEVICE UCB

end UCBDEF9;

end_module \$UCBDEF;

MODULE \$TTYUCBDEF;

```

/*
/* $TTYUCBDEF follows here only because there is no way to get the
/* UCBSK_LENGTH symbol into another module. TTYUCBDEF was formerly
/* included in TTYDEF.MAR.
/*
/* TERMINAL DRIVER DEFINITIONS
/*
/* These definitions define the device dependent extensions of the UCB.
/* Certain portions of the ucb are assumed to be contiguous and must not
/* be split. These areas are documented in the following definitions.
/*

aggregate TTYUCBDEF structure prefix UCBS;

    TT_UCBFILL byte dimension #UCB_LENGTH fill prefix UCBDEF tag $$;

/*
/* Logical terminal UCB extension
/*

    TL_CTRLY      longword unsigned; /* CONTROL Y AST BLOCK LIST HEAD
    TL_CTRLC      longword unsigned; /* CONTROL C AST BLOCK LIST HEAD
    TL_OUTBAND     longword unsigned; /* OUT OF BAND CHARACTER MASK
    TL_BANDQUE     longword unsigned; /* OUT OF BAND AST QUEUE
    TL_PHYUCB      longword unsigned; /* THE PHYSICAL UCB ADDRESS
    TL_CTLPID      longword unsigned; /* CONTROLLING PID (USED WITH SPAWN)
    TL_BRKTHRU     quadword unsigned; /* FACILITY BROADCAST BITMASK

    constant TL_LENGTH equals . tag C;
    constant TL_LENGTH equals . tag K;

/*
/* Terminal class driver dependant region
/* Split here between local and remote terminal UCB's
/*

    TTYRTTUCB union; /* local/remote union (overlay)

    TTYUCB structure; /* this structure defines remainder of local ucb

/* READ TIMEOUT CONTROL

    TT_RDUE      longword unsigned; /* ABSTIME WHEN READ TIMEOUT DUE
    TT_RTIMOU     longword unsigned; /* ADDRESS OF READ TIMEOUT ROUTINE

/* TERMINAL DRIVER STATE TABLE

    TT_STATE_OVERLAY union fill;
    TT_STATE      quadword unsigned; /* CURRENT UNIT STATE VECTOR
    TT_STATE_Q_BLOCK structure fill;
    TT_STATE_T_OVERLAY union fill;
    TT_STATE1      longword unsigned;
    TT_STATE1_FIELDS structure fill prefix TTY$;
        ST_POWER      bitfield mask; /*
        ST_CTRL      bitfield mask; /*
        ST_FILL        bitfield mask; /*
        ST_CURSOR      bitfield mask; /*
        ST_SENDF        bitfield mask; /*

```

```

    ST_BACKSPACE    bitfield mask; /*
    ST_MULTI         bitfield mask; /*
    ST_WRITE         bitfield mask; /*
    ST_EOL           bitfield mask; /*
    ST_EDITREAD      bitfield mask; /*
    ST_RDVERIFY      bitfield mask; /*
    ST_RECALL        bitfield mask; /*
    ST_READ          bitfield mask; /*
end TT_STATE1_FIELDS;
end TT_STATE1_OVERLAY;

TT_STATE2_OVERLAY union fill;
TT_STATE2 longword unsigned;
TT_STATE2_FIELDS structure fill prefix TTYS;
    ST_CTRLO         bitfield mask; /*
    ST_DEL           bitfield mask; /*
    ST_PASALL        bitfield mask; /*
    ST_NOECHO        bitfield mask; /*
    ST_WRTALL        bitfield mask; /*
    ST_PROMPT        bitfield mask; /*
    ST_NOFLTR        bitfield mask; /*
    ST_ESC           bitfield mask; /*
    ST_BADESC        bitfield mask; /*
    ST_NL            bitfield mask; /*
    ST_REFRSH        bitfield mask; /*
    ST_ESCAPE        bitfield mask; /*
    ST_TYPFUL        bitfield mask; /*
    ST_SKIPLF        bitfield mask; /*
    ST_ESC_O         bitfield mask; /*
    ST_WRAP          bitfield mask; /*
    ST_OVRFLO        bitfield mask; /*
    ST_AUTOP         bitfield mask; /*
    ST_CTRLR         bitfield mask; /*
    ST_SKIPCRLF      bitfield mask; /*
    ST_EDITING       bitfield mask; /*
    ST_TABEXPAND      bitfield mask; /*
    ST_QUOTING        bitfield mask; /*
    ST_OVERSTRIKE    bitfield mask; /*
    ST_TERMORM       bitfield mask; /*
    ST_ECHAES        bitfield mask; /*
    ST_PRE           bitfield mask; /*
    ST_NINTMULTI     bitfield mask; /*
    ST_RECONNECT     bitfield mask; /*
    ST_CTSLOW        bitfield mask; /*
    ST_TABRIGHT      bitfield mask; /*
end TT_STATE2_FIELDS;
end TT_STATE2_OVERLAY;
end TT_STATE_0_BLOCK;
end TT_STATE_OVERLAY;

TT_LOGUCB          longword unsigned; /* ADDRESS OF THE LOGICAL UCB

/* DEFAULT CHARACTERISTICS

TT_DECHAR          longword unsigned; /* DEFAULT DEVICE CHARACTERISTICS
TT_DECHA1          longword unsigned; /* DEFAULT DEVICE CHAR EXTENSIONS
```

/* WRITE QUEUE POINTERS

```

      TT_WFLINK      longword unsigned;    /* Write queue forward link.
      TT_WBLINK      longword unsigned;    /* Write queue backward link.
      TT_WRTBUF       longword unsigned;    /* Current write buffer block.
/* ADDRESS AND LENGTH OF MULTI-ECHO STRING
      TT_MULTI        longword unsigned;    /* CURRENT MULTIECHO BUFFER ADDRESS
      TT_MULTILEN      word unsigned;       /* LENGTH OF STRING TO OUTPUT
      TT_SMLTLEN       word unsigned;       /* SAVED MULTI LENGTH
      TT_SMLT          longword unsigned;   /* AND THE SAVED ADDRESS
/* Typeahead buffer address

```

```

      TT_TYPAHD       longword unsigned;    /* TYPEAHEAD BUFFER ADDRESS

```

```

/*-- *****

```

/* DEFAULT SPEED, FILL, PARITY (MUST BE CONTIGUOUS)

```

/*++ *****

```

```

      TT_DESPEE       word unsigned;        /* DEFAULT SPEED
      TT_DECRF         byte unsigned;       /* DEFAULT CR FILL
      TT_DELFF         byte unsigned;       /* DEFAULT LF FILL

      TT_DEPAR1        byte unsigned;       /* DEFAULT PARITY/CHAR SIZE
      TT_DEFSPE_SPARE1 byte unsigned;
      TT_DEFSPE_SPARE2 word unsigned;

```

```

/*-- *****

```

/* DEFAULT TERMINAL TYPE AND SIZE (MUST BE CONTIGUOUS)

```

/*
/*++ *****

```

```

      TT_DETTYPE       byte unsigned;       /* DEFAULT TERMINAL TYPE
      TT_DESIZE         word unsigned;      /* DEFAULT LINE SIZE
      TT_SPARE1         byte unsigned;      /* SPARE BYTE MUST FOLLOW

```

```

/*-- *****

```

/* SPEED, FILL, PARITY (MUST BE CONTIGUOUS)

```

/*++ *****

```

```

      TT_SPEED_OVERLAY union fill;
      TT_SPEED         word unsigned;      /* SPEED CODES (SPLIT SPEED)
      TT_SPEED_FIELDS structure fill;
      TT_TSPEED         byte unsigned;     /* TRANSMIT SPEED
      TT_RSPEED         byte unsigned;     /* RECEIVE SPEED
      end TT_SPEED_FIELDS;
end TT_SPEED_OVERLAY;

```

```

      TT_CRFILL        byte unsigned;      /* NUMBER FILLS TO OUTPUT ON CR
      TT_LFFILL        byte unsigned;      /* NUMBER FILLS TO OUTPUT ON LF

```

```

      PARITY_OVERLAY union fill;

```

```

        TT_PARITY    byte unsigned;          /* PARITY AND CHARACTER SIZE DEFINITIONS
        TT_PARITY BITS structure fill;
        TT_XXPARITY   bitfield mask;         /* UNUSED ??
        TT_DISPARERR  bitfield mask;         /* SPECIFY DISREGARD PARITY ERRORS
        TT_USERFRAME  bitfield mask;         /* SPECIFY USER FRAME SETUP
        TT_LEN         bitfield mask length 2; /* CHARACTER LENGTH
        TT_STOP       bitfield mask;         /* STOP BITS
        TT_PARTY      bitfield mask;         /* PARITY ENABLED
        TT_ODD        bitfield mask;         /* ODD PARITY
    end TT_PARITY BITS;
end PARITY_OVERLAY;
TT_PAR_SPARE1    byte unsigned;
TT_PAR_SPARE2    word unsigned;
/*-- *****

/* CURRENT CURSOR AND LINE POSITION FOR FORMATTED OPERATIONS

    TT_CURSOR      word unsigned;           /* CURRENT CURSOR POSITION
    TT_LINE         byte unsigned;          /* CURRENT LINE ON PAGE
    TT_LASTC        byte unsigned;          /* LAST FORMATTED OUTPUT CHARACTER

/* Number of back spaces to output for non-ansi terminals

    TT_BSLEN        word unsigned;          /* NUMBER OF BACKSPACES

/* FILL HANDLING

    TT_FILL         byte unsigned;          /* CURRENT FILL COUNT

/* ESCAPE SYNTAX RULE STATE.

    TT_ESC          byte unsigned;          /* CURRENT READ ESCAPE SYNTAX STATE
    TT_ESC_O        byte unsigned;          /* OUPUT ESCAPE STATE

/* Count of characters in interrupt string

    TT_INTCNT       byte unsigned;

/* Bit used for modem control

    TT_UNITBIT      word unsigned;          /* BIT USED TO ENABLE AND DISABLE MODEM CONTROL.

/* PORT SPECIFIC OUTPUT CONTROL

    TT_HOLD_OVERLAY union fill;
        TT_HOLD word unsigned;             /* UNIT HOLDING TANK AND PORT DISPATCH
        TT_HOLD BITS structure fill prefix TTYS;
            TANK_CHAR    byte unsigned;     /* CHARACTER
            TANK_PREMPT   bitfield mask;    /* SEND PREMPT CHARACTER
            TANK_STOP     bitfield mask;    /* STOP OUTPUT
            TANK_HOLD     bitfield mask;    /* CHAR IN TANK
            TANK_BURST    bitfield mask;    /* BURST ACTIVE
            TANK_DMA      bitfield mask;    /* DMA ACTIVE **** SHOULD MOVE BEFORE BURST ****
    end TT_HOLD BITS;
end TT_HOLD_OVERLAY;

```

```

TT_PREMPT      byte unsigned;      /* THE BYTE USED TO PREMPT INPUT
TT_OUTTYPE     byte unsigned;      /* TYPE OF OUTPUT THAT THIS CALL
/* CLASS & PORT VECTOR POINTERS
TT_GETNXT      longword unsigned;  /* ADDRESS OF CLASS INPUT ROUTINE
TT_PUTNXT      longword unsigned;  /* ADDRESS OF CLASS OUTPUT ROUTINE
TT_CLASS       longword unsigned;  /* ADDRESS OF CLASS VECTOR
TT_PORT        longword unsigned;  /* ADDRESS OF PORT VECTOR

TT_OUTADR      longword unsigned;  /* ADDRESS OF OUTPUT CURRENT STREAM
TT_OUTLEN      word unsigned;      /* LENGTH OF OUTPUT STREAM

TT_PRTCTL_OVERLAY union fill;
  TT_PRTCTL     word unsigned;      /* THE PORT DRIVER CONTROL WORD
  TT_PRTCTL_BITS structure fill prefix TTYS;
    PC_NOTIME   bitfield mask;      /* IF SET NO TIMEOUT WILL BE CALCULATED
    PC_DMAENA   bitfield mask;      /* DMA CURRENTLY ENABLED
    PC_DMAAVL   bitfield mask;      /* DMA SUPPORTED ON THIS PORT
    PC_PRMMAP   bitfield mask;      /* UNIT CAN HAVE PERMANENT MAP REGISTERS
    PC_MAPAVL   bitfield mask;      /* MAP REGISTER CURRENTLY ALLOCATED
    PC_XOFAVL   bitfield mask;      /* AUTO XOFF SUPPORTED ON THIS PORT
    PC_XOFENA   bitfield mask;      /* AUTO XOFF CURRENTLY ENABLED
    PC_NOCRLF   bitfield mask;      /* don't do free linefeed after creturn
  end TT_PRTCTL_BITS;
end TT_PRTCTL_OVERLAY;

/* MODEM CONTROL DEFINITIONS
TT_DS_RCV      byte unsigned;      /* CURRENT RECEIVE MODEM
TT_DS_TX       byte unsigned;      /* CURRENT TRANSMIT MODEM
TT_DS_ST       word unsigned;      /* CURRENT MODEM STATE
TT_DS_TIM      word unsigned;      /* CURRENT MODEM TIMEOUT

TT_MAINT_OVERLAY union fill;
  TT_MAINT      byte unsigned;      /* MAINTENANCE PARAMETERS
  TT_MAINT_BITS structure fill;
    TT_MAINT_FILL bitfield length 7;
    TT_DSBL     bitfield mask;      /* LINE DISABLED
  end TT_MAINT_BITS;
end TT_MAINT_OVERLAY;

TT_OLDCPZORG   byte unsigned;      /* spare byte make this longword alligned

constant TT_CLSLEN equals . tag C;
constant TT_CLSLEN equals . tag K;

/*****
/*
/* Terminal Port driver dependant extension region
TP_MAP         longword unsigned;  /* UNIBUS MAP REGISTERS
TP_STAT_OVERLAY union fill;
  TP_STAT       byte unsigned;      /* DMA PORT SPECIFIC STATUS
  TP_STAT_BITS structure fill prefix TTYS; /* BITS DEFINED IN THE DMA STATUS WORD
  TP_ABORT      bitfield mask;      /* DMA ABORT REQUESTED ON THIS LINE

```

```

        TP_ALLOC bitfield mask;          /* ALLOC MAP FORK IN PROGRESS
        TP_DLOC bitfield mask;          /* DEALLOCATE MAP FORK IN PROGRESS
    end TP_STAT BITS;
end TP_STAT_OVERLAY;

TP_SPARE1      byte unsigned;
TP_SPARE2      word unsigned;

constant TP_LENGTH equals . tag C;
constant TP_LENGTH equals . tag K;
constant TT_LENGTH equals . tag C;
constant TT_LENGTH equals . tag K;

TT_STATE SX structure prefix TTYS;
    SX_POWER      bitfield; /*
    SX_CTRL      bitfield; /*
    SX_FILL      bitfield; /*
    SX_CURSOR      bitfield; /*
    SX_SENDF      bitfield; /*
    SX_BACKSPACE  bitfield; /* OUTPUT BACKSPACES FOR SEVERAL LOOPS
    SX_MULTI      bitfield; /*
    SX_WRITE      bitfield; /* Write state
    SX_EOL        bitfield; /*
    SX_EDITREAD   bitfield; /*
    SX_RDVERIFY   bitfield; /*
    SX_RECALL     bitfield; /*
    SX_READ       bitfield; /*
    SX_FILLBITS   bitfield length 32-^; /* END OF FIRST LONGWORD

    SX_CTRL0      bitfield; /*
    SX_DEL        bitfield; /*
    SX_PASALL     bitfield; /*
    SX_NOECHO     bitfield; /*
    SX_WRTALL     bitfield; /*
    SX_PROMPT     bitfield; /*
    SX_NOFLTR     bitfield; /*
    SX_ESC        bitfield; /*
    SX_BADESC     bitfield; /*
    SX_NL         bitfield; /* New line must directly precede
    SX_REFRSH     bitfield; /* refresh, or all breaks.
    SX_ESCAPE     bitfield; /*
    SX_TYPFUL     bitfield; /*
    SX_SKIPLF     bitfield; /*
    SX_ESC O      bitfield; /*
    SX_WRAP       bitfield; /*
    SX_OVRFLO     bitfield; /*
    SX_AUTOP      bitfield; /*
    SX_CTRLR      bitfield; /*
    SX_SKIPCRLF   bitfield; /*
    SX_EDITING     bitfield; /*
    SX_TABEXPAND   bitfield; /*
    SX_QUOTING     bitfield; /*
    SX_OVERSTRIKE bitfield; /*
    SX_TERMORM     bitfield; /*
    SX_ECHAES      bitfield; /*
    SX_PRE         bitfield; /*

```

```

        SX_NINTMULTI    bitfield;    /*
        SX_RECONNECT    bitfield;    /*
        SX_CTSLOW        bitfield;    /*
        SX_TABRIGHT      bitfield;    /*
    end TT_STATE_SX;
end TTYUCB;

```

```
/* remote terminal extension
```

```
RTTUCB structure;
```

```

RTT_NETUCB        longword unsigned; /* NET DEVICE UCB
RTT_NETWIND        longword unsigned; /* NET DEVICE WCB
RTT_IRPFL          longword unsigned; /* IRP QUEUE
RTT_IRPBL          longword unsigned; /* IRP QUEUE
RTT_NETIRP         longword unsigned; /* READ NET IIRP
RTT_BANDINCL       longword unsigned; /* OUT OF BAND INCLUDES
RTT_BANDINMSK      longword unsigned; /* OUT OF BAND INCLUDE MASK
RTT_BANDEXCL       longword unsigned; /* out of band exclude mask
RTT_BANDEXMSK      longword unsigned; /* out of band exclude

RTT_PROVRS        byte unsigned;      /* PROTOCOL VERSION
RTT_PROECO        byte unsigned;      /* PROTOCOL ECO
RTT_LINK          word unsigned;      /* LINK NUMBER (for LOGINOUT)

RTT_OBJ           byte unsigned;      /* OBJECT NUMBER CONNECTED
RTT_SYSTYPE       word unsigned;      /* SYSTEM TYPE (VMS=7)
RTT_FILLBYTE      byte unsigned;      /* fill - use when convenient

```

```
/* CTERM driver only
```

```

CT_FLAGS_OVERLAY union fill;
    CT_FLAGS        word unsigned;    /* MISC FLAGS
    CT_FLAGS_BITS structure fill prefix "FLGS";
        WIIRP_BSY    bitfield mask;  /* WIIRP BUSY
        CTRL0        bitfield mask;  /* CTRL0 IN PROGRESS
        CANCTRL0     bitfield mask;  /* CANCEL CTRL0 ON WRITE
        INWRTFDT     bitfield mask;  /* IN WRITE FDT
        QUOTA        bitfield mask;  /* QUOTA CHARGED
        VAXTOVAX     bitfield mask;  /* VAX TO VAX
        BUFFER       bitfield mask;  /* DO BUFFERED WRITES
    end CT_FLAGS_BITS;
end CT_FLAGS_OVERLAY;

```

```

CT_QCTPCNT        word unsigned;      /* QUEUED CTP COUNT
CT_WIIRP          longword unsigned;  /* WRITE IIRP
CT_TQE           longword unsigned;  /* TQE ADDRESS
CT_NETQFL        longword unsigned;  /* WAITING FOR WRITE
CT_NETQBL        longword unsigned;  /* TO NET QUEUE
CT_STALLQFL      longword unsigned;  /* IRPs BEING HELD
CT_STALLQBL      longword unsigned;  /* QUEUE
CT_WRTCTP        longword unsigned;  /* BUFFERED WRITE CTP
CT_WRTCUR        longword unsigned;  /* CURRENT FILL POINTER
CT_WRTSIZ        word unsigned;      /* REMAINING SIZE
CT_WRTCNT        word unsigned;      /* COUNT SINCE LAST TQE
CT_MAXMSG        word unsigned;      /* MAX WRITE TO NET SIZE

```

CT_MAXREAD	word unsigned;	/* MAX READ IN SERVER
CT_LEGALMSG	longword unsigned;	/* LEGAL MESSAGE MASK
CT_VERSION	byte unsigned;	/* CTERM VERSION
CT_ECO	byte unsigned;	/* CTERM ECO
CT_FILLWORD	word unsigned;	/* fill

CT_DEBUG_FILL	CHARACTER LENGTH 4*10;	/* 10 LONGWORD FOR DEBUG
---------------	------------------------	--------------------------

constant RTT_LENGTH equals . tag C;	/* Length must be same for both RTTDRIVER
constant RTT_LENGTH equals . tag K;	/* and CTDRIVER.

end RTTUCB;

end TTYRTTUCB; /* end union

end TTYUCBDEF;

end_module \$TTYUCBDEF;

```
module $VADEF;
```

```
/*+  
/* VIRTUAL ADDRESS VIELDS  
/*-
```

```
aggregate VADEF union prefix VAS;  
  VADEF_BITS0 structure fill;  
    'BYTE' bitfield mask length 9;          /*BYTE VIELD  
    VPN bitfield mask length 21;            /*VIRTUAL PAGE NUMBER  
    P1 bitfield mask;                       /*P1 SPACE  
    SYSTEM bitfield mask;                  /*SYSTEM SPACE  
  end VADEF_BITS0;  
  VADEF_BITS1 structure fill;  
    FILL 1 bitfield length 9 fill prefix VADEF tag $$;  
    VPG bitfield mask length 23;           /*VIRTUAL PAGE INCLUDING P1 & S  
  end VADEF_BITS1;  
end VADEF;  
end_module $VADEF;
```

```
module $VCADEF;
```

```
/**
```

```
/*
```

```
/* VCA - Volume Cache Block. This block contains the specialized caches for
/* a disk volume; to wit, the file ID cache, the extent cache, and the quota
/* file cache. The file ID cache and extent cache are together in one block;
/* the quota cache is located separately in another block. Both are pointed to
/* by the VCB.
```

```
/*
```

```
/*-
```

```
aggregate VCADEF structure prefix VCA$;
```

```
  FIDCACHE longword unsigned; /* pointer to file ID cache
  EXTCACHE longword unsigned; /* pointer to extent cache
  SIZE word unsigned; /* block size
  TYPE byte unsigned; /* block type code
  FLAGS structure byte; /* cache flags
    FIDC_VALID bitfield mask; /* FID cache valid
    EXTC_VALID bitfield mask; /* Extent cache valid
    FIDC_FLUSH bitfield mask; /* FID cache to be flushed
    EXTC_FLUSH bitfield mask; /* Extent cache to be flushed
  end FLAGS;
  constant 'LENGTH' equals . tag K; /* length of block header
  constant 'LENGTH' equals . tag C; /* length of block header
```

```
/*
/* The file ID cache consists of the cache header, followed by a longword
/* vector of file numbers, densely packed.
```

```
/*
```

```
end VCADEF;
```

```
aggregate VCADEF1 structure prefix VCA$;
```

```
  FIDSIZE word unsigned; /* number of entries allocated
  FIDCOUNT word unsigned; /* number of entries present
  FIDCLKID longword unsigned; /* FID cache lock id.
  FIDCACB byte unsigned dimension 28; /* FID cache blocking ACB
  FIDLIST longword unsigned; /* first entry in list
```

```
/*
/* The extent cache consists of the cache header, followed by a quadword
/* vector of extents, densely packed. Each quadword contains block count
/* and starting LBN.
```

```
/*
```

```
end VCADEF1;
```

```
aggregate VCADEF2 structure prefix VCA$;
```

```
  EXTSIZE word unsigned; /* number of entries allocated
  EXTCOUNT word unsigned; /* number of entries present
  EXTTOTAL longword unsigned; /* total number of blocks contained in cache
  EXTLIMIT word unsigned; /* limit of volume to be cached, in percent/10
  FILL_2 word fill tag $$; /* spare
  EXTCCLKID longword unsigned; /* EXT cache lock id.
  EXTCACB byte unsigned dimension 28; /* Extent cache blocking ACB.
  EXTLIST quadword unsigned; /* first entry in list
```

```
end VCADEF2;
```

```

aggregate VCADEF3 structure prefix VCAS;
    EXTBLOCKS longword unsigned;          /* number of blocks
    EXTLBN longword unsigned;              /* starting LBN
/*
/* The quota cache consists of the cache header, followed by the cache
/* entries. Each cache entry is a block as defined below.
/*
end VCADEF3;

aggregate VCADEF4 structure prefix VCAS;
    QUOSIZE word unsigned;                 /* number of entries allocated
    QUOLRU word unsigned;                  /* current LRU counter
    QUOCLKID longword unsigned;            /* whole cache lock ID
    FILL 3 byte dimension 3 fill tag $$;   /* 2nd longword & block size & type
    QUOCLFLAGS structure byte;             /* cache flags
        CACHEVALID bitfield mask;         /* cache is valid
        CACHEFLUSH bitfield mask;         /* cache is to be flushed
    end QUOCACHEFLAGS;
    QUOACB byte unsigned dimension 28;     /* ACB to deliver blocking AST
    QUOFLUSHACB byte unsigned dimension 28; /* ACB to deliver cache flush AST
    QUOLIST longword unsigned;             /* start of entries

end VCADEF4;

aggregate VCADEF5 structure prefix VCAS;
    QUOLOCK structure;                    /* lock status block
        QUOSTATUS OVERLAY union fill;
            QUOSTATUS word unsigned;       /* $ENQ status
            QUOINDEX word unsigned;        /* index in cache of this entry
        end QUOSTATUS_OVERLAY;
        QUOLRUX word unsigned;             /* LRU index for entry
        QUOLKID longword unsigned;         /* lock ID of cache entry
        QUORECNUM byte unsigned dimension 3 tag L; /* record number
        QUOFLAGS structure byte unsigned;  /* flags byte
            QUOVALID bitfield mask;        /* valid entry is present
            QUODIRTY bitfield mask;        /* dirty flag
        end QUOFLAGS;
        USAGE longword unsigned;          /* current usage
        PERMQUOTA longword unsigned;       /* permanent quota
        OVERDRAFT longword unsigned;       /* overdraft limit
    end QUOLOCK;
    QUOUIC longword unsigned;              /* UIC
    constant QUOLENGTH equals . tag K;     /* length of quota cache entry
    constant QUOLENGTH equals . tag C;     /* length of quota cache entry
end VCADEF5;

end_module $VCADEF;

```

```
{+
VCB - VOLUME CONTROL BLOCK
```

```
{ THERE IS ONE VOLUME CONTROL BLOCK FOR EACH MOUNTED DEVICE UNIT IN A
{ SYSTEM. IT CONTAINS INFORMATION NECESSARY TO CONTROL ACCESS TO AND
{ VERIFY CERTAIN VOLUME PARAMETERS IN THE CASE A DEVICE UNIT SHOULD
{ ERRONEOUSLY GO OFFLINE.
{-
```

```
module $VCBDEF;
```

```
aggregate VCBDEF_COMMON structure prefix VCB$;
```

```
  FORWARD_LINK union fill;
    FCBFL longword unsigned;
    BLOCKFL longword unsigned;
    MEMQFL longword unsigned;
  end FORWARD_LINK;
  BACKWARD_LINK union fill;
    FCBBL longword unsigned;
    BLOCKBL longword unsigned;
    MEMQBL longword unsigned;
  end BACKWARD_LINK;
  SIZE word unsigned;
  TYPE byte unsigned;
```

```
  constant MRKLEN equals .;
  constant MRKLEN equals : tag C;
  #VCBMARK2 = .;
```

```
  VOLSTS union fill;
    STATUS byte unsigned;
    DISK BITS structure fill;
      WRITE_IF bitfield mask;
      WRITE_SM bitfield mask;
      HOMBLRBAD bitfield mask;
      IDXHDRBAD bitfield mask;
      NOALLOC bitfield mask;
      EXTFID bitfield mask;
      GROUP bitfield mask;
      SYSTEM bitfield mask;
    end DISK BITS;
    TAPE BITS structure fill;
      PARTFILE bitfield mask;
      LOGICEOVS bitfield mask;
      WAIMOUVOL bitfield mask;
      WAIREWIND bitfield mask;
      WAIUSRLBL bitfield mask;
      CANCELIO bitfield mask;
      MUSTCLOSE bitfield mask;
      NOWRITE bitfield mask;
    end TAPE BITS;
    SHADOW BITS structure fill;
      SHADMAST bitfield mask;
      NEWSSMEMB bitfield mask;
      FAILED bitfield mask;
    end SHADOW BITS;
```

```
( COMMON VCB DEFINITIONS
```

```
/* FCB listhead forward link
/* or - Blocked request listhead forward link
/* or - Shadow set members queue forward link
```

```
/* FCB listhead backward link
/* or - Blocked request listhead backward link
/* or - Shadow set members queue backward link
```

```
/* Size of VCB in bytes
/* structure type of VCB
```

```
/* Mark length
/* Mark length
/* Second mark point
```

```
/* Volume status:
( for disks:
/* Index file is write accessed
/* Storage map is write accessed
/* Primary home block is bad
/* Primary index file header is bad
/* Allocation/deallocation inhibited (bad bitmaps)
/* Volume has 24 bit file numbers
/* Volume is mounted /group
/* Volume is mounted /system
```

```
( for tapes:
/* Partial file exists on tape
/* Positioned at logical end of volume set
/* Wait for volume mount
/* Wait for rewind completion
/* Wait for user label
/* Cancel I/O
/* Must close file
/* Don't write trailers
```

```
( for shadow set members
/* This VCB is for shadow set master
/* New shadow set member
/* Member failed out of shadow set
```

```

end VOLSTS;
TRANS word unsigned;
RVN word unsigned;
AQB longword unsigned;
VOLNAME character length 12;
RVT longword unsigned;
#VCBMARK3 = .;
constant COMLEN equals . prefix VCB$ tag K;
end VCBDEF_COMMON;

```

```

aggregate VCBDEF_DISKS structure prefix VCB$;
filldisks byte dimension #VCBMARK3 fill;
constant COMLEN equals . prefix VCB$ tag C;

```

```

HOMELBN longword unsigned;
HOME2LBN longword unsigned;
IXHDR2LBN longword unsigned;
IBMAPLBN longword unsigned;
SBMAPLBN longword unsigned;
IBMAPSIZE byte unsigned;
SBMAPSIZE byte unsigned;
IBMAPVBN byte unsigned;
SBMAPVBN byte unsigned;
CLUSTER word unsigned;
EXTEND word unsigned;
FREE longword unsigned;
MAXFILES longword unsigned;
WINDOW byte unsigned;
LRU LIM byte;
FILEPROT word unsigned;
MOUNT word unsigned;
EOFDELTA byte unsigned;
RESFILES byte unsigned;
RECORDSZ word unsigned;
BLOCKFACT byte unsigned;
STATUS2 OVERLAY union fill;
  STATUS2 byte unsigned;
  STATUS2 BITS structure fill;
    WRITETHRU bitfield;
    NOCACHE bitfield;
    MOUNTVER bitfield;
    ERASE bitfield;
    NOHIGHWATER bitfield;
    NOSHARE bitfield;
    CLUSLOCK bitfield;
  end STATUS2 BITS;
end STATUS2_OVERLAY;

```

```

QUOTAFCB longword unsigned;
CACHE longword unsigned;
QUOCACHE longword unsigned;
QUOSIZE word unsigned;
PENDERR word unsigned;
SERIALNUM longword unsigned;
JNLIOCNT longword unsigned;
RETAINMIN quadword unsigned;
RETAINMAX quadword unsigned;
VOLLKID longword unsigned;

```

```

/* VOLUME TRANSACTION COUNT
/* RELATIVE VOLUME NUMBER
/* ADDRESS OF AQB
/* VOLUME LABEL BLANK FILLED
/* ADDRESS OF UCB OR RELATIVE VOLUME TABLE
/* THIRD MARK POINT
/* LENGTH OF COMMON AREA

```

```

/* LENGTH OF COMMON AREA
/* LBN OF VOLUME HOME BLOCK
/* LBN OF ALTERNATE VOLUME HOME BLOCK
/* LBN OF ALTERNATE INDEX FILE HEADER
/* LBN OF INDEX FILE BITMAP
/* LBN OF STORAGE BITMAP
/* SIZE OF INDEX FILE BITMAP
/* SIZE OF STORAGE BITMAP
/* CURRENT VBN IN INDEX FILE BIT MAP
/* CURRENT VBN IN STORAGE MAP
/* VOLUME CLUSTER SIZE
/* VOLUME DEFAULT FILE EXTENSION LENGTH
/* NUMBER OF FREE BLOCKS ON VOLUME
/* MAXIMUM NUMBER OF FILES ALLOWED ON VOLUME
/* VOLUME DEFAULT WINDOW SIZE
/* VOLUME DIRECTORY LRU SIZE LIMIT
/* VOLUME DEFAULT FILE PROTECTION
/* MOUNT COUNT
/* INDEX FILE EOF UPDATE COUNT
/* NUMBER OF RESERVED FILES ON VOLUME
/* NUMBER OF BYTES IN A RECORD
/* VOLUME BLOCKING FACTOR

```

```

/* SECOND STATUS BYTE

```

```

/* VOLUME IS TO BE WRITE-THROUGH CACHED
/* ALL CACHEING IS DISABLED ON VOLUME
/* VOLUME CAN UNDERGO MOUNT VERIFICATION
/* ERASE DATA WHEN BLOCKS REMOVED FROM FILE
/* TURN OFF HIGH-WATER MARKING (D = ON)
/* non-shared mount
/* CLUSTER WIDE LOCKING NECESSARY

```

```

/* ADDRESS OF FCB OF DISK QUOTA FILE
/* ADDRESS OF VOLUME CACHE BLOCK
/* ADDRESS OF VOLUME QUOTA CACHE
/* LENGTH OF QUOTA CACHE TO ALLOCATE
/* COUNT OF PENDING WRITE ERRORS
/* VOLUME SERIAL NUMBER (DISKS ONLY)
/* JOURNALING IO COUNT
/* MINIMUM FILE RETENTION PERIOD
/* MAXIMUM FILE RETENTION PERIOD
/* VOLUME LOCK ID

```

```

VOLCKNAM character length 12;
BLOCKID longword unsigned;
MOUNTTIME quadword unsigned;
MEMHDFL longword unsigned;
MEMHDBL longword unsigned;
ACTIVITY word unsigned;
fill_1 byte fill;
SHAD_STS byte unsigned;
SHAD_RESV longword unsigned;
ACB byte unsigned dimension 28;
MIN_CLASS structure;
    FILL_2 byte dimension 20 fill;
end MIN_CLASS;
MAX_CLASS structure;
    FILL_3 byte dimension 20 fill;
end MAX_CLASS;
constant "LENGTH" equals . prefix VCB$ tag K;
constant "LENGTH" equals . prefix VCB$ tag C;

end VCBDEF_DISKS;

/*
/* SHADOW SET MEMBER VOLUME CONTROL BLOCK FIELDS
/*

aggregate VCBDEF_SHADOW structure prefix VCB$;
    fillshadow byte dimension #VCBMARK3 fill;
    MEM_UCB longword unsigned;
    MAST_UCB longword unsigned;
    MAST_VCB longword unsigned;
    WORKQFL longword unsigned;
    WORKQBL longword unsigned;
    MSCP_STS longword unsigned;
    SHDM_RESV quadword unsigned;
    constant SHAD_LEN equals .;
end VCBDEF_SHADOW;

/*
/* MTAACP VOLUME CONTROL BLOCK FIELDS
/*

aggregate VCBDEF2 structure prefix VCB$;
    FILL_3 byte dimension #VCBMARK3 fill prefix VCBDEF tag $$;
    CUR_FID_OVERLAY union fill;
        CUR_FID longword unsigned;
        CUR_FID_FIELDS structure fill;
            CUR_NUM word unsigned;
            CUR_SEQ word unsigned;
        end CUR_FID_FIELDS;
    end CUR_FID_OVERLAY;
    START_FID_OVERLAY union fill;
        START_FID longword unsigned;
        START_FID_FIELDS structure fill;
            START_NUM word unsigned;
            START_SEQ word unsigned;
        end START_FID_FIELDS;

```

```

/* NAME FOR VOLUME LOCKS
/* VOLUME BLOCKING LOCK.
/* VOLUME MOUNT TIME
/* SHADOW SET MEMBERS QUEUE HEADER FL
/* SHADOW SET MEMBERS QUEUE HEADER BL
/* ACTIVITY COUNT/FLAG

/* STATUS BYTE RELATIVE TO MEMHDFL
/* RESERVED FOR SHADOW SET PROCESSING
/* ACB FOR BLOCKING AST.
/* MINIMUM CLASSIFICATION

/* MAXIMUM CLASSIFICATION

/* LENGTH OF STANDARD VCB
/* LENGTH OF STANDARD VCB

/* Shadow set member UCB address
/* Shadow set master UCB address
/* Shadow set master VCB address
/* Work queue forward link
/* Work queue backward link
/* MSCP status information
/* Reserved for future enhancements
/* Shadow set member VCB length

/* CURRENT FILE IDENTIFICATION
/* CURRENT FILE SECTION NUMBER
/* CURRENT FILE SEQUENCE NUMBER

/* FILE IDENTIFICATION AT START OF SEARCH
/* FILE SECTION NUMBER AT START OF SEARCH
/* FILE SEQUENCE NUMBER AT START OF SEARCH

```

```

end START_FID_OVERLAY;
MODE_OVERLAY union fill;
  MODE word unsigned;
  MODE_BITS structure fill;
    OVREXP bitfield;
    OVRACC bitfield;
    OVRLBL bitfield;
    OVRSETID bitfield;
    INTCHG bitfield;
    EBCDIC bitfield;
    NOVOL2 bitfield;
    NOHDR3 bitfield;
    STARFILE bitfield;
    ENUSEREOT bitfield;
    BLANK bitfield;
    INIT bitfield;
    NOAUTO bitfield;
    OVRVOL0 bitfield;
    FIL_ACCESS bitfield;
  end MODE_BITS;
end MODE_OVERLAY;

```

/* MODE OF OPERATION

```

/* OVERRIDE EXPIRATION
/* OVERRIDE ACCESS
/* OVERRIDE LABELS
/* OVERRIDE SET IDENTIFIER
/* INTERCHANGE TAPE
/* EBCDIC CODE SET
/* DO NOT WRITE A VOL2 LABEL
/* DO NOT WRITE HDR3 LABELS
/* CURRENT FILE IS A STARLET PRODUCED FILE
/* SET WHEN USER HANDLING OF EOT IS ENABLED
/* SET FOR AVL WHEN NO READ SHOULD HAPPEN FIRST
/* SET FOR AVL WHEN NEXT VOL MOUNTED SHOULD BE INITED
/* MTAACP NOT RUNNING IN AVL AND AVR MODE
/* OVERRIDE THEVOL1 OWNER IDENT FIELD
/* SET IF ACCESS ROUTINE ALLOWS CHECK OF VMS PROTECTION ON FILE

```

```

end MODE_OVERLAY;
TM byte unsigned;
CUR_RVN byte unsigned;
ST_RECORD longword unsigned;
MVC longword unsigned;
WCB longword unsigned;
VPFL longword unsigned;
VPBL longword unsigned;
USRLBLAST longword unsigned;
LBLCNT byte unsigned;

```

```

/* NUMBER OF TM'S INTO FILE
/* CURRENT RELATIVE VOLUME
/* NUMBER OF RECORDS UP TO AND INCLUDING LAST TAPE MARK
/* ADDRESS OF MAGNETIC TAPE VOLUME LIST
/* ADDRESS OF WINDOW FOR THIS VOLUME
/* VIRTUAL PAGE LIST HEAD
/* VIRTUAL PAGE LIST TAIL
/* ADDRESS OF USER LABEL AST CONTROL BLOCK
/* Count of HDRn labels read on file open

```

/* NOTE THAT FCP AND MTAACP SHARE VCB\$W_MCOUNT(DISPLACEMENT 76)

end VCBDEF2;

aggregate VCBDEF3 structure prefix VCB\$;

FILL 4 byte dimension #VCBMARK2 fill prefix VCBDEF tag \$\$;

QNAMECNT byte unsigned;

QNAME character length 20;

/* BYTE COUNT OF QUEUE NAME

/* ASCII NAME OF QUEUE FOR THIS DEVICE

/* JOURNAL ACP VOLUME CONTROL BLOCK FIELDS

end VCBDEF3;

aggregate VCBDEF4 structure prefix VCB\$;

FILL 5 byte dimension #VCBMARK3 fill prefix VCBDEF tag \$\$;

JNL_CHAR_OVERLAY union fill;

JNL_CHAR longword unsigned;

JNL_CHAR_BITS structure fill;

JNL_DISK bitfield mask;

JNL_TAPE bitfield mask;

JNL_TMPFI bitfield mask;

end JNL_CHAR_BITS;

end JNL_CHAR_OVERLAY;

JNL_JFTA longword unsigned;

/* journal media characteristics

/* journal is on disk

/* journal is on tape

/* temporary file

/* JOURNAL FILE TABLE ADDRESS (IN ACP)

```
JNL_IRPS longword unsigned dimension 2;
JNL_JMT longword unsigned;
JNL_UCB longword unsigned;
JNL_JMTFL longword unsigned;
JNL_JMTBL longword unsigned;
JNL_MODE byte unsigned;
JNL_COP word unsigned;
FILE 2 byte fill prefix VCBDEF tag $$;
JNL_MASK longword unsigned;
constant JNL_LENGTH equals . prefix VCB$ tag K;
constant JNL_LENGTH equals . prefix VCB$ tag C;
end VCBDEF4;

/* PREALLOCATED FREE IRP QUEUE HEADER
/* ADDRESS OF JMT (JOURNAL MERGE TABLE)
/* UCB ADDRESS
/* JMT FORWARD LINK
/* JMT BACKWARD LINK
/* ACCESS MODE OF CREATOR
/* NUMBER OF JOURNAL FILE COPIES
/* SPARE
/* MASK
/* LENGTH OF JOURNAL VCB
/* LENGTH OF JOURNAL VCB

end_module $VCBDEF;
```

```
module $VL1DEF;
```

```
/*+  
/* VOL1 ANSI MAGNETIC TAPE LABEL  
/* THIS IS THE FIRST BLOCK ON EVERY ANSI LABELED MAGNETIC TAPE.  
/* IT IDENTIFIES THE VOLUME AND ITS PROTECTION.  
/*-
```

```
aggregate VL1DEF structure prefix VL1$;
```

```
  VL1LID longword unsigned;          /*LABEL IDENTIFIER AND NUMBER 'VOL1'  
  VOLLBL character length 6;         /*VOLUME LABEL  
  VOLACCESS byte unsigned;           /*VOLUME ACCESS  
  FILL 1 character length 13 fill prefix VL1DEF tag $$; /*SPACES  
  SYSCODE character length 13;       /* SYSTEM CODE  
  OWNER UNION union fill;  
    OWNER IDENT character length 14; /* VOL1 OWNER ID FIELD  
    OLD VOLOWNER structure fill;  
      VOLOWNER character length 13; /*VOLUME OWNER IDENTIFICATION  
      DECSTDVER byte unsigned;      /*DEC STANDARD VERSION  
    end OLD VOLOWNER;  
  end OWNER UNION;  
  FILL 2 character length 28 fill prefix VL1DEF tag $$; /*SPACES  
  LBLSTDVER byte unsigned;           /*LABEL STANDARD VERSION '3'  
end VL1DEF;
```

```
end_module $VL1DEF;
```

```
module $VL2DEF;
```

```
/*+
```

```
/* VOL2 ANSI MAGNETIC TAPE LABEL
```

```
/* THIS IS BLOCK IS WRITTEN TO TAPES WHEN A VMS PROTECTION IS SPECIFIED
```

```
/*-
```

```
aggregate VL2DEF structure prefix VL2$;
```

```
    VL2LID longword unsigned;
```

```
    VOOWNER character length 15;
```

```
end VL2DEF;
```

```
end_module $VL2DEF;
```

```
/*LABEL IDENTIFIER AND NUMBER 'VOL2'  
/*VOLUME OWNER IDENTIFICATION
```

```
module $WCBDEF;
```

```
/*+
/* WCB - WINDOW CONTROL BLOCK
/*
/* THERE IS A WINDOW CONTROL BLOCK FOR EACH FILE ACCESSED BY A PROCESS.
/* IT CONTAINS MAPPING INFORMATION SUCH THAT A LARGE PERCENTAGE OF VIRTUAL
/* FILE I/O CAN BE MAPPED FROM VIRTUAL TO LOGICAL BLOCK NUMBERS WITHOUT
/* HAVING TO READ THE RESPECTIVE FILE HEADER.
/*-
```

```
aggregate WCBDEF structure prefix WCB$;
```

```

WFL longword unsigned; /* WINDOW LIST FORWARD LINK
WBL longword unsigned; /* WINDOW LIST BACKWARD LINK
SIZE word unsigned; /* SIZE OF WINDOW BLOCK IN BYTES
TYPE byte unsigned; /* STRUCTURE TYPE OF WCB
ACCESS OVERLAY union fill;
  ACCESS byte unsigned; /* ACCESS CONTROL BYTE
  ACCESS BITS structure fill;
    READ bitfield mask; /* READ ACCESS ALLOWED (1=YES)
    WRITE bitfield mask; /* WRITE ACCESS ALLOWED (1=YES)
    NOTFCP bitfield mask; /* FILE NOT ACCESSED BY FCP IF SET
    SHRWCB bitfield mask; /* SHARED WINDOW
    OVERDRAWN bitfield mask; /* FILE ACCESSOR HAS OVERDRAWN HIS QUOTA
    COMPLETE bitfield mask; /* SET WINDOW MAPS ENTIRE FILE
    CATHEDRAL bitfield mask; /* LARGE, COMPLEX WINDOW (SIC) TO MAP
    EXPIRE bitfield mask; /* FILE COMPLETELY
    /* FILE EXPIRATION DATE MAY NEED TO BE SET
  end ACCESS BITS;
end ACCESS OVERLAY;
PID OVERLAY union fill;
  PID longword unsigned; /* PROCESS ID OF ACCESSOR PROCESS
  PID_FIELDS structure fill;
    /* FILL 5 byte dimension 2 fill prefix WCBDEF tag $$;
    REFCNT word unsigned; /* REFERENCE COUNT FOR SHARED WINDOW
  end PID_FIELDS;
end PID OVERLAY;
ORGUCB longword unsigned; /* ADDRESS OF ORIGINAL UCB FROM CCB
ACON OVERLAY union fill;
  ACON word unsigned; /* ACCESS CONTROL INFORMATION
  /* NOTE - THESE BITS TRACK THE BITS
  /* IN FIB$$_ACCTL
  ACON_BITS0 structure fill;
    ROWRITE bitfield; /* NO OTHER WRITERS
    DLOCK bitfield; /* ENABLE DEACCESS LOCK
    /* FILL 1 bitfield length 2 fill prefix WCBDEF tag $$; /* UNUSED
    SPOOC bitfield; /* SPOOL FILE ON CLOSE
    WRITECK bitfield; /* ENABLE WRITE CHECK
    SEQONLY bitfield; /* SEQUENTIAL ONLY ACCESS
    /* FILL 2 bitfield fill prefix WCBDEF tag $$; /* SPARE
    WRITEAC bitfield; /* WRITE ACCESS
    READCK bitfield; /* ENABLE READ CHECK
    NOREAD bitfield; /* NO OTHER READERS
    NOTRUNC bitfield; /* NO TRUNCATES
  end ACON_BITS0;
```

```

ACON BITS1 structure fill;
  FILL 3 bitfield length 2 fill prefix WCBDEF tag $$;
  /* THE FOLLOWING FIELD OVERLAYS THE FIRST
  /* UNUSED FLAG IN WCB$W_ACON ABOVE.
  NOACCLOCK bitfield; /* NO ACCESS LOCK CHECKING
  FILL 4 bitfield length 8 fill prefix WCBDEF tag $$;
  READINIT bitfield; /* A READINIT WAS DONE OVER THIS CHANNEL
  WRITE TURN bitfield; /* FORCE WINDOW TURN ON WRITES
end ACON BITS1;
end ACON OVERLAY;
#WCBMARK2 = . ; /* SIZE DEVICE INDEPENDENT PART OF WCB
NMAP word unsigned; /* NUMBER OF MAPPING POINTERS
FCB longword unsigned; /* ADDRESS OF FCB
RVT longword unsigned; /* ADDRESS OF RELATIVE VOLUME TABLE
LINK longword unsigned; /* LINK TO NEXT WINDOW SEGMENT
READS longword unsigned; /* COUNT OF READS PERFORMED
WRITES longword unsigned; /* COUNT OF WRITES PERFORMED
STVBN longword unsigned; /* STARTING VBN MAPPED BY WINDOW
constant MAP equals . prefix WCB$ tag K; /* MAP POINTERS START HERE
constant MAP equals . prefix WCB$ tag C; /* MAP POINTERS START HERE
constant 'LENGTH' equals . prefix WCB$ tag K; /* LENGTH OF STANDARD WCB SANS POINTERS
constant 'LENGTH' equals . prefix WCB$ tag C; /* LENGTH OF STANDARD WCB SANS POINTERS
/* NOTE THAT VIRTUAL MAPPING
/* NEEDS P1 COUNT IMMEDIATELY
/* FOLLOWING STVBN
/* COUNT FIELD OF FIRST POINTER
/* LBN FIELD OF SECOND POINTER
/* COUNT FIELD OF SECOND POINTER
/* LBN FIELD OF FIRST POINTER
/* FORMAT OF RETRIEVAL POINTER

P1_COUNT word unsigned;
P1_LBN longword unsigned;
P2_COUNT word unsigned;
P2_LBN longword unsigned;

end WCBDEF;

aggregate WCBDEF1 structure prefix WCB$;
  COUNT word unsigned; /* COUNT FIELD
  LBN longword unsigned; /* LBN FIELD
end WCBDEF1;

aggregate WCBDEF2 structure prefix WCB$ origin FILL_6;
  PREVCOUNT word unsigned; /* PREVIOUS RETRIEVAL POINTER
  PREVLBN longword unsigned; /* RETRIEVAL POINTER FORMAT
  FILL 6 byte fill prefix WCBDEF tag $$;
end WCBDEF2;

aggregate WCBDEF3 structure prefix WCB$;
  FILL 7 byte dimension #WCBMARK2 fill prefix WCBDEF tag $$;
  JNL_REFC word unsigned; /* REFERENCE COUNT AT $ASSJNL TIME
  JNL_FCOD word unsigned; /* FACILITY CODE OWNER JOURNAL CHANNEL
  JNL_STAT OVERLAY union fill ;
    JNL_STAT byte unsigned; /* STATUS
    JNL_STAT BITS structure fill ;
      JDB Bitfield mask ; /* JDB ALLREADY WRITTEN OVER THIS CHANNEL
    end JNL_STAT BITS ;
  end JNL_STAT OVERLAY ;
  JNL_ACMOD byte unsigned; /* ACCESS MODE
  JNL_PROT word unsigned; /* PROTECTION MASK

```

```
JNL_AID longword unsigned;
JNL_SEQ longword unsigned;
JNL_PRCNAM character length 16;
JNL_UIC longword unsigned;
JNL_PUIC longword unsigned;
JNL_TIME quadword unsigned;
JNL_WLFL longword unsigned;
JNL_WLBL longword unsigned;
JNL_RC longword unsigned;
constant JNL_LEN equals . prefix WCB$ tag K;
constant JNL_LEN equals . prefix WCB$ tag C;
end WCBDEF3;

end_module $WCBDEF;
```

```
/* ASSIGN SEQUENCE NUMBER
/* SEQUENCE NUMBER LAST ENTRY WRITTEN
/* PROCESS NAME OF CHANNEL OWNER
/* UIC USED FOR ENTRIES WRITTEN OVER CHANNEL
/* UIC OF PROCESS
/* TIME AT WHICH CHANNEL WAS ASSIGNED
/* FORWARD LINK WCB QUEUE
/* BACKWARD LINK WCB QUEUE
/* READ CONTEXT BLOCK
/* LENGTH WCB FOR JOURNAL CHANNELS
/* LENGTH WCB FOR JOURNAL CHANNELS
```

```
module $WSLDEF;
```

```
/*
```

```
/* WORKING SET LIST DEFINITIONS
```

```
/*
```

```
aggregate WSLDEF union prefix WSL$;
```

```
  WSLDEF BITS structure fill;
```

```
    VACID bitfield mask;
```

```
    PAGTYP bitfield mask length 3;
```

```
    PFNLOCK bitfield mask;
```

```
    WSLOCK bitfield mask;
```

```
    GOODPAGE bitfield mask;
```

```
    FILL 1 bitfield fill prefix WSLDEF tag $$;
```

```
    MODIFY bitfield mask;
```

```
end WSLDEF_BITS;
```

```
/*WSL ENTRY VALID
```

```
/*PAGE TYPE (SEE PFNDEF FOR VALUES)
```

```
/*PAGE FRAME LOCK
```

```
/*THE PRECEDING 5 BITS MUST BE IN ORDER
```

```
/*WORKING SET LOCK
```

```
/*THIS PAGE SHOULD REMAIN IN WS ONE MORE PASS
```

```
/*SPARE BIT
```

```
/*SAVED MODIFY BIT
```

```
/*THE FOLLOWING 5 BITS MUST BE IN ORDER
```

```
  constant 'LENGTH' equals 4 prefix WSL tag $C; /*SIZE OF WS LIST ENTRY
```

```
/*
```

```
/* PAGE TYPE VIELD DEFINITIONS
```

```
/*
```

```
/* N.B.: These constants have been adjusted by left-shifting the constant by the offset to the field WSL$V_PAGTYP.
```

```
/*
```

```
/*
```

```
/*
```

```
/*
```

```
  IF .wsle [wsl$V_pagtyp] EQL (wsl$C_system*-1)
```

```
  ! Or (wsl$C_system/2)
```

```
  constant PROCESS equals %X00 prefix WSL tag $C; /*PROCESS PAGE
```

```
  constant SYSTEM equals %X02 prefix WSL tag $C; /*SYSTEM PAGE
```

```
  constant 'GLOBAL' equals %X04 prefix WSL tag $C; /*GLOBAL PAGE (READ ONLY)
```

```
  constant GBLWRT equals %X06 prefix WSL tag $C; /*GLOBAL WRITABLE PAGE
```

```
  constant PPGTBL equals %X08 prefix WSL tag $C; /*PROCESS PAGE TABLE
```

```
  constant GPGTBL equals %X0A prefix WSL tag $C; /*GLOBAL PAGE TABLE
```

```
end WSLDEF;
```

```
end_module $WSLDEF;
```

```
module $WQHDEF;
```

```
/*+
```

```
/* WAIT QUEUE HEADER DEFINITIONS
```

```
/*-
```

```
aggregate WQHDEF structure prefix WQHS;
```

```
WQFL longword unsigned;
```

```
WQBL longword unsigned;
```

```
WQCNT word unsigned;
```

```
WQSTATE word unsigned;
```

```
constant 'LENGTH' equals . prefix WQHS tag K;
```

```
constant 'LENGTH' equals . prefix WQHS tag C;
```

```
/*HEAD OR FORWARD LINK
```

```
/*TAIL OR BACKWARD LINK
```

```
/*WAIT QUEUE COUNT
```

```
/*STATE NUMBER FOR WAIT
```

```
/*LENGTH OF WAIT QUEUE HEADER
```

```
/*LENGTH OF WAIT QUEUE HEADER
```

```
end WQHDEF;
```

```
end_module $WQHDEF;
```

```
{+
{ XG - Definitions for the fields within the XGDRIVER.
{-
```

```
module $XGDEF;
constant PRI_XMT equals 0 prefix XG tag $C; /* Primary xmt use vector slot 0
constant SEC_XMT equals 1 prefix XG tag $C; /* Secondary xmt use vector slot 1
constant PRI_RCV equals 2 prefix XG tag $C; /* Primary rcv use vector slot 2
constant SEC_RCV equals 3 prefix XG tag $C; /* Secondary rcv use vector slot 3
constant RCV_CSR equals 0 prefix XG tag $C; /* Receive CSR
constant XMT_CSR equals 2 prefix XG tag $C; /* Transmit CSR
constant MISC_REG equals 4 prefix XG tag $C; /* Set misc bits
constant IND_ADDR equals 6 prefix XG tag $C; /* Use to access the ind reg (IR)
constant(
    PROTOCOL /* 0th IR def's the protocol char
    , RCV_ERR /* 1st IR def's rcv errors
    , XMT_ERR /* 2nd IR def's xmt errors
    , SYNC /* 3rd IR def's sync characteristics
    , MODEM /* 4th IR def's modem state change
    , STN_ADDR /* 5th IR use to set station address
    , PRI_RCV /* 6th and 7th IR used to define
    , PRI_RCV1 /* primary rcv buffer and address
    , SEC_RCV /* 8th and 9th IR used to define
    , SEC_RCV1 /* secondary rcv buffer and address
    , PRI_XMT /* 10th and 11th IR used to define
    , PRI_XMT1 /* primary xmt buffer and address
    , SEC_XMT /* 12th and 13th IR used to define
    , SEC_XMT1 /* secondary xmt buffer and address
    , TERM_CHAR /* 14th used to describe term char
    , FREE /* 15th unused register
) equals 0 increment 1 prefix XG tag $C;
```

```
/* Bit def's for RCV and XMT CSR
```

```
aggregate XGDEF union fill prefix XG$;
    XGDEF BITSO structure fill;
    ENABLE bitfield mask; /* Enable the receiver
    FILL_1 bitfield fill prefix XGDEF tag $$; /* reserved
    PRM_SEC bitfield mask; /* 0 = prim 1 = sec buffer and addr
    TERM_IDL bitfield mask; /* Term char for RCV's Idle for XMT's
    DATA_SET_IE bitfield mask; /* Enable intrpts for data set change
    INT_ENABLE bitfield mask; /* Enable intrpts for rcv and xmt's
    ACT_DSC bitfield mask; /* Active (rcv's) Data set change (xmt')
    DONE_S bitfield mask; /* Sec buffer processing is finished
    ILP_XCS bitfield mask; /* Internal loopback (rcv) XMT clock src
    LOOP_TYPE bitfield mask length 2; /* Loopb type for devices like CPI which support many
    FILL_2 bitfield fill prefix XGDEF tag $$; /* reserved
    RESIDUAL bitfield mask; /* Bit protocols only
    PRI_SEC_STN bitfield mask; /* 0 = control 1 = tributary station
    ERROR bitfield mask; /* Error on rcv or xmt
    DONE_P bitfield mask; /* Primary buffer processing complete
```

end XGDEF_BITS0;

/* Misc reg definitions

```
XGDEF BITS1 structure fill;
  IND_REG bitfield length 4;          /* Ind reg address to access
  FILE_3 bitfield length 3 fill prefix XGDEF tag $$; /* reserved
  MASTER_RESET bitfield mask;        /* Master reset bit
  FILL_4 bitfield length 2 fill prefix XGDEF tag $$; /* reserved
  USER_RCV_FLAG bitfield mask;       /* User receive flag
  FILL_5 bitfield fill prefix XGDEF tag $$;
  CTS_FLAG bitfield mask;            /* Clear to send flag
  CARRIER_FLAG bitfield mask;       /* Carrier detect flag
  RING_FLAG bitfield mask;           /* Ring indicator flag
  DSR_FLAG bitfield mask;            /* Data set ready flag
end XGDEF_BITS1;
```

/* Protocol parameter definitions Indirect register 0

```
XGDEF BITS2 structure fill;
  ERR_CNTRL bitfield length 3;        /* Error control def CRC CCITT 1's
  PROTOCOL bitfield length 3;        /* Protocol type def DDCMP
  STRIP_SYNC bitfield mask;          /* Set to strip excess sync characters
  FILL_6 bitfield fill prefix XGDEF tag $$; /* reserved
  RCV_BPC bitfield length 3;         /* RCV bits/char default is 8
  FILE_7 bitfield length 2 fill prefix XGDEF tag $$; /* reserved
  XMT_RCV bitfield length 3;         /* XMT bits/char default is 8
end XGDEF_BITS2;
```

/* Receive errors definitions Indirect register 1

```
XGDEF BITS3 structure fill;
  FILL_8 bitfield fill prefix XGDEF tag $$; /* reserved
  LATENCY_RCV bitfield mask;         /* RCV latency error
  NXM_RCV bitfield mask;             /* Non-existent memory error
  BCC_ERR bitfield mask;             /* Block check error
  VRC_ERR bitfield mask;             /* Byte prot only char parity error
  ABORT bitfield mask;              /* Bit prot only
  BUFOVR bitfield mask;             /* When char COUNT and msg len aren't eq
  FILL_9 bitfield fill prefix XGDEF tag $$; /* Reserved
  RES_BIT_CNT bitfield mask length 3; /* Residual bit count
  FILE_10 bitfield length 5 fill prefix XGDEF tag $$; /* Reserved
end XGDEF_BITS3;
```

/* Transmit error definitions Indirect register 2

```
XGDEF BITS4 structure fill;
  MSG_LEN bitfield mask;             /* Char count indicates a buff too small
  NXM_XMT bitfield mask;             /* Non-existent memory
  LATENCY_XMT bitfield mask;        /* XMT latency error
  FILL_11 bitfield length 5 fill prefix XGDEF tag $$; /* Reserved
  XMT_BRG bitfield length 4;        /* Baud rate
  FILE_12 bitfield length 4 fill prefix XGDEF tag $$; /* Reserved
end XGDEF_BITS4;
```

/* Sync information definitions Indirect register 3

```
XGDEF BITS5 structure fill;
  NMB_OF_SYNC bitfield length 5; /* Number of syncs to send betweenmsgs
  FILL_13 bitfield length 3 fill prefix XGDEF tag $$; /* Reserved
  SYNC bitfield length 8; /* Contains the sync char
end XGDEF_BITS5;
```

/* Data set change register Indirect register 4

```
XGDEF BITS6 structure fill;
  FILL_14 bitfield length 4 fill prefix XGDEF tag $$; /* Reserved
  CTS bitfield mask; /* Clear to send
  CARRIER bitfield mask; /* Carrier detect
  RING_IND bitfield mask; /* Ring indicator
  DSR bitfield mask; /* Data set ready
  USER_XMT bitfield mask; /* User transmit
  DTR bitfield mask; /* Data terminal ready
  DATA_SGNL bitfield mask; /* Data signal rate
  FILL_15 bitfield fill prefix XGDEF tag $$; /* reserved
  RTS bitfield mask; /* Request to send
  FILL_16 bitfield length 3 fill prefix XGDEF tag $$; /* Reserved
end XGDEF_BITS6;
```

/* Internal clock def's TX.CSR<8>

```
constant INTCLK_OFF equals 0 prefix XG tag $C; /* No internal clock
constant INTCLK_ON equals 1 prefix XG tag $C; /* Set internal clock
```

/* Error control definitions IRO<0:3>

```
constant(
  ERR_CRC1 /* CRC-CCITT preset to 1's
  , ERR_CRC0 /* CRC-CCITT preset to 0's
  , ERR_LVE /* LRC/VRC even
  , ERR_CRC16 /* CRC-16 preset to 0's
  , ERR_LR0 /* LRC odd
  , ERR_LRCE /* LRC even
  , ERR_LVO /* LRC/VRC odd
  , NOCON /* No error control
) equals 0 increment 1 prefix XG tag $C;
```

/* Protocol definitions IRO<3:3>

```
constant PRO_DDCMP equals 0 prefix XG tag $C; /* DDCMP
constant PRO_SDLC equals 1 prefix XG tag $C; /* SDLC
constant PRO_HDLC equals 2 prefix XG tag $C; /* HDLC
constant BISYNC equals 3 prefix XG tag $C; /* BISYNC
constant GENBYTE equals 7 prefix XG tag $C; /* General byte
```

/* Bits per char definitions. RCV: IRO<8:10> XMT: IRO<13:15>

```
constant(
  BPC_8
```

```

      . BPC_1
      . BPC_2
      . BPC_3
      . BPC_4
      . BPC_5
      . BPC_6
      . BPC_7
    ) equals 0 increment 1 prefix XG tag $C;

```

```
/* Baud rate generator definitions IR2<8:11>
```

```

constant(
  BRG_800
  . BRG_1200
  . BRG_1760
  . BRG_2152
  . BRG_2400
  . BRG_4800
  . BRG_9600
  . BRG_19200
) equals 0 increment 1 prefix XG tag $C;

```

```
/* Sync character definitions IR3<8:15>
```

```

constant SYNC_DDCMP equals 150 prefix XG tag $C; /* Set sync character to HEX 96
constant SYNC_HDLC equals 0 prefix XG tag $C; /* Set no sync character
constant SYNC_BISYNC equals 50 prefix XG tag $C; /* Set sync character to HEX 32

```

```
/* Struct of parameter buffer
```

```
end XGDEF;
```

```
aggregate XGDEF1 structure fill prefix XG$;
```

```
ERR_CNTRL byte unsigned;
```

```
PROTOCOL byte unsigned;
```

```
TX_BPC byte unsigned;
```

```
RX_BPC byte unsigned;
```

```
BAUD byte unsigned;
```

```
NUM_SYNC byte unsigned;
```

```
SYNC_REG byte unsigned;
```

```
ICLK byte unsigned;
```

```
BPC byte unsigned;
```

```
MNTLOOP_OVERLAY union fill;
```

```
  MNTLOOP byte unsigned;
```

```
/* Set the type of error control to use
```

```
/* Set protocol type
```

```
/* Set XMT bits per char
```

```
/* Set RCV bits per char
```

```
/* Set line speed
```

```
/* Set number of sync to send
```

```
/* Set sync char to send
```

```
/* Set the internal clock
```

```
/* RCV/XMT bits per char
```

```
/* Maint loopb type
```

```
/*
```

```
/* Bit def for interface with the frame routine
```

```
/*
```

```
/* XG$V_BUFFER_CHAR clear Buffer char in the next position
```

```
/* set Use XG$V_BUFFER_IN_PREV_POS
```

```
/* XG$V_BUFFER_IN_PREV_POS
```

```
/* clear ignore the char
```

```
/* set Buffer in previous position
```

```
/* XG$V_COMPLETE_READ set complete framed buffer to user
```

```
/*
  MNTLOOP BITS structure fill;
    BUFFER_CHAR bitfield mask;
    BUFFER-IN PREV POS bitfield mask;
    COMPLETE_READ bitfield mask;
    FILL 17 bitfield length 28 fill prefix XGDEF tag $$; /* reserved
    NEW_FRAME bitfield mask; /* set if new rcv message
  end MNTLOOP BITS;
end MNTLOOP_OVERLAY;
end XGDEF1;

end_module $XGDEF;
```

CMO

O

N

B_E

EXA

EXE

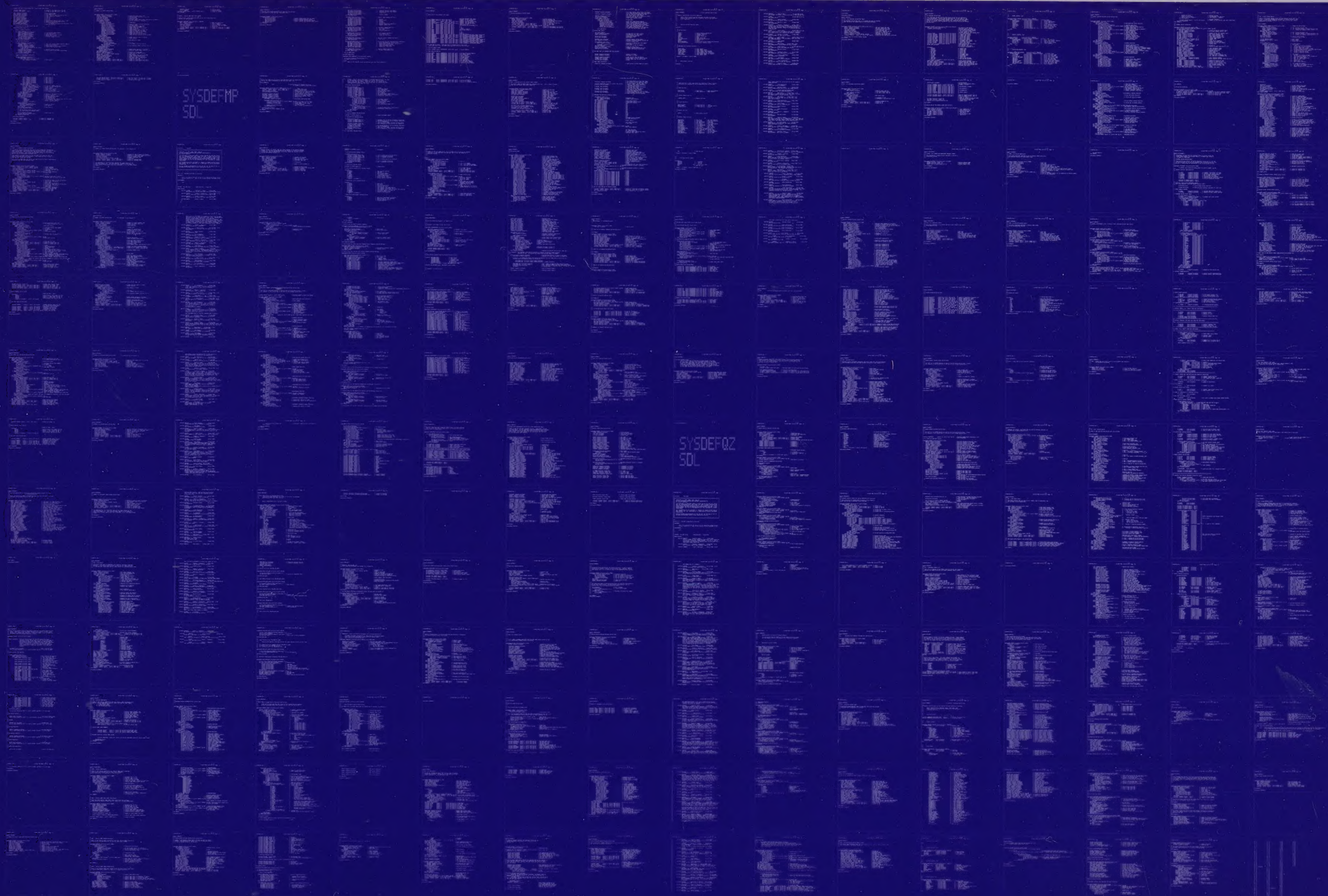
EXI

EXE

EXE

0371 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



0372

AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY